This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

THIS PAGE BLANK (USPTO)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 December 2002 (19.12.2002)

PCT

(10) International Publication Number
WO 02/102097 A1

(51) International Patent Classification[7]: H04Q 7/20, G06F 17/60

(21) International Application Number: PCT/US02/18069

(22) International Filing Date: 7 June 2002 (07.06.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/877,414 8 June 2001 (08.06.2001) US

(71) Applicant: SEURAT COMPANY [US/US]; 5718 Central Avenue, Boulder, CO 80301 (US).

(72) Inventors: BUUS, Bryan; 12581 Maria Circle, Broomfield, CO 80020 (US). GAZ, Randall; 245 Remuda Lane, Lafayette, CO 80026 (US). MALAHER, Tom; 5016 Norquay Drive NW, Calgary, Alberta T2K 2L3 (CA). MULLER, Michael; Unit D, 848 Walnut Street, Boulder, CO 80302 (US). SCHWARTZ, Keith; 3625 N. Seeley Avenue, Chicago, IL 60618 (US).

(74) Agent: COOLEY GODWARD LLP; Patent Group, 11951 Freedom Drive, One Freedom Square-Reston Town Center, Reston, VA 20190-5656 (US).

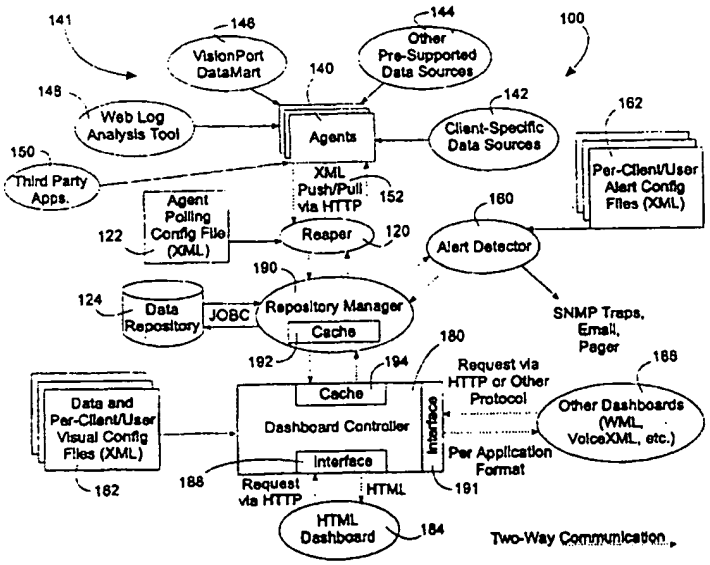
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR MONITORING KEY PERFORMANCE INDICATORS IN A BUSINESS



(57) Abstract: A system for monitoring business performance indicators is disclosed for a network. The system comprises a data source having a predefined format with an agent (140) communicating with the data source where the agent translates the data into modified data. There is also a reaper coupled to the agent which takes the modified data and a data repository coupled to the reaper and configured to store the data and an alert detector coupled to the data repository and configured to compare the modified data with a configuration parameter and a dashboard controller coupled to the data repository and configured to display the modified data in a format defined by a second configuration parameter.

WO 02/102097 A1

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEM AND METHOD FOR MONITORING KEY PERFORMANCE INDICATORS IN A BUSINESS

FIELD OF THE INVENTION

The present invention pertains to systems and methods for measuring, quantifying, and monitoring business performance and operations, and more particularly, to systems and methods for analyzing key business metrics in a business environment and for providing a visual snapshot of the business metrics specified by a user.

BACKGROUND OF THE INVENTION

The availability and maintenance of an electronic business system is rapidly becoming the rule rather than the exception in order for a company to compete in today's marketplace. Interactive web-sites allow companies to reach a wide range of audiences and to actively promote and sell their products, services, or other offerings. However, with the integration of an interactive and dynamic web-site or other on-line business system comes the need to provide for continuous monitoring of the performance and utilization parameters of these automated business systems.

Systems for performing basic statistical analysis of raw business data are known and available in various forms. For instance, the analysis of traditional business performance indicators such as sales volume, profit margins, inventory levels, order placement, and revenue levels are readily calculated from a database compilation of the relevant information. Known systems are adapted to query a database that contains the raw data used to calculate these performance indicators and are adapted to provide the resulting information to a user in various formats. Similarly, known systems and products that merely provide data mining or data warehousing do not allow the flexibility to incorporate analysis systems from third parties or the ability to interrelate the corresponding business metrics with ongoing business operations.

In the context of an electronic commerce-based business environment, access to dynamic information such as system load time, web-site visits ("page hits"), registered users, page refresh rate, and product ordering demographics is also required in order to effectively monitor and evaluate the performance of a particular electronic business system or web-site. When provided to a user, this information is often needed in a dynamic form in order to accurately portray the business metrics associated with the data. However, traditional statistical analysis systems or database applications are not conducive to monitoring information that is constantly being updated or is by its very nature dynamic. Further, the analysis of electronic business metrics often involves complicated query structures and logic algorithms that are not easily replicated on a static system such as a relational database. Known third party applications that specialize in analyzing a particular type of information are often better suited and more efficient at analyzing and reporting this type of information but known systems are unable to effectively integrate these types of third party analysis tools into a monitoring system structured for a particular company.

Furthermore, known software applications and other systems fail to provide a user with a business metrics monitoring platform that allows real-time integration with strategic performance indicators while also being able to integrate existing legacy data systems and third party applications. Known systems also fail to provide usable information beyond generic statistics and cannot quantify the impact of specific business initiatives on an overall business objective.

SUMMARY OF THE INVENTION

In one aspect, a system for monitoring business performance indicators in a networked environment, comprises a data source having a predefined format, an agent communicatively coupled to the data source, wherein the agent is configured according to the data source format and wherein the agent is operative to gather data from the data source and translate the data into a first modified format thereby creating modified data. The system also comprises a reaper communicatively coupled to the agent and configured to retrieve the modified data from the agent, a data repository

communicatively coupled to the reaper and configured to store the modified data an alert detector communicatively coupled to the data repository and configured to compare the modified data with a first configuration parameter, and a dashboard controller communicatively coupled to the data repository and configured to display the modified data in a format defined by a second configuration parameter.

In another aspect, a system for monitoring business performance indicators in a networked environment, comprises a data source having a predefined format, an agent communicatively coupled to the data source, wherein the agent is configured according to the data source format and wherein the agent is operative to gather data from the data source and translate the data into a first modified format thereby creating modified data. The system also comprises a reaper communicatively coupled to the agent and configured to retrieve the modified data from the agent, a repository manager communicatively coupled to the reaper, a data repository communicatively coupled to the repository manger, an alert detector communicatively coupled to the repository manager, and a dashboard controller communicatively coupled to the repository manager.

In another aspect, a method for monitoring a business metric in a networked environment, comprises coupling to a data source having a known format, wherein the data source includes data that represents the business metric, configuring an agent according to the data source format, gathering the data from the data source via the agent, translating the data into a first modified format, storing the modified data in a data repository, comparing the modified data with an alert parameter range, displaying the modified data in a format defined by a second configuration parameter, determining whether the modified data falls within the alert parameter range, and producing an alert if the modified data falls within the alert parameter range.

In yet another aspect, a system for monitoring a business metric in a networked environment, comprises a processor, a data storage device, and an instruction set residing on the data storage device, wherein the instruction set is configured to perform a method comprising coupling to a data source having a known format, wherein the

data source includes data that represents the business metric, configuring an agent according to the data source format, gathering the data from the data source via the agent, translating the data into a first modified format, storing the modified data in a data repository, comparing the modified data with an alert parameter range, displaying the modified data in a format defined by a second configuration parameter, determining whether the modified data falls within the alert parameter range, and producing an alert if the modified data falls within the alert parameter range.

In a further aspect, a system for monitoring business performance indicators in a networked environment, comprises a collector adapted to communicatively coupled to a data source having a predetermined format, wherein the collector is configured according to the data source format and wherein the collector is operative to gather data from the data source and translate the data into a first modified format thereby creating modified data, a data manager communicatively coupled to the collector and configured to manage the input and output of the modified data between the collector and a data storage device, wherein the data manager is adapted to communicatively couple with an alert device, and a display interface communicatively coupled to the data manager and configured to display the modified data in a format defined by a second configuration parameter.

As will become apparent to those skilled in the art, numerous other embodiments and aspects of the invention will become evident hereinafter from the following descriptions and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

The drawings illustrate both the design and utility of the preferred embodiments of the present invention, wherein:

Figure 1 is an architecture diagram of a dashboard system constructed in accordance with an embodiment of the present invention;

Figure 2 is an architecture diagram of a dashboard system constructed in accordance with another embodiment of the present invention;

Figure 3 is a screenshot of a dashboard in accordance with an embodiment of the present invention;

Figures 4 - 10 are various details of the dashboard of Figure 3; and

Figures 11 and 12 are sample dashboards for an organization utilizing an association-franchise relationship.

DETAILED DESCRIPTION

Figure 1 shows a general architectural diagram of a dashboard system 100 constructed in accordance with an embodiment of the present invention. In a preferred embodiment, the dashboard system 100 is one module of an overall business monitoring and reporting system that is utilized to present a user with an overview of business metrics and other key performance indicators. The dashboard will sometimes be referred to herein as the VisionPort™ Dashboard System. VisionPort™ is a trademark of XOR, Inc.

Generally, the dashboard system 100 provides a high level executive view of key performance indicators for an eBusiness. The dashboard system 100 preferably presents a visually oriented snapshot of the status of key business metrics identified by a user. Preferably, the dashboard system 100 is an open and extensible framework of computer software or fixed hardware components designed to enable organizations to build real-time measurement and monitoring solutions for applications and systems. The dashboard can be installed at a user's own facility or maintained at an off-site Application Services Provider for a particular client. Additionally, the various components of the dashboard system 100 may be spread across different platforms or physical locations.

The dashboard system 100 is preferably configurable on a per-user or per-client basis, customizable to show each user or client exactly what information they want access to. For instance, a marketing group can be shown one set of information such as

sales and advertising revenues while a company officer can be shown a broader scope of information about the performance of the company. In addition to being able to gather information from legacy or other existing proprietary company information, the dashboard system 100 is adapted to either link to or direct a user to an appropriate 3rd party application tool to get the required information. The dashboard system 100 is preferably viewable through readily available internet browser software, such as current versions of Internet Explorer® or Netscape Navigator®. Other devices such as WAP-compatible phones, Palm OS based devices, Blackberry, VoiceXML, and email programs are also preferably compatible with the information provided by the dashboard system 100 so that a user can access the information from a variety of interface devices. In that respect, the framework of the dashboard system 100 is preferably extensible in order to support a wide range of application interfaces.

The diagram depicted in Figure 1 provides an overview of the technical architecture of a preferred dashboard system framework and its system requirements. The dashboard system 100 includes four primary components, a reaper 120, one or more agents 140, an alert detector 160, and a dashboard controller 180. While each of these four primary components will be described in more detail below, the following is an overview of their function and structure. Briefly, the agents 140 are used to gather information from various data sources, third party applications, data servers, legacy systems, etc., and reformat the data into a common scheme using XML. The reaper 120 is adapted to contact the agents in order to gather the reformatted data and store the reformatted data in a data repository (reference number 124 in Figure 1). The reaper 120 is also preferably adapted to perform data aggregation and history creation. A repository manager 190 is also shown in Figure 1 that further manages the inflow and outflow of data to and from the data repository 124.

The alert detector 160 reads the data from the repository and compares the information to an alert configuration parameter. If an alert is detected, the alert detector initiates an appropriate action such as an SNMP trigger, paging, email, or visual alert. The dashboard controller 180 acts as the interface between an end-user and the information in the data repository. The dashboard controller requests information from

the data repository 124 and converts the requested information into a format compatible with the user's selected interface. The dashboard controller 180 is also adapted to read a configuration file in order to determine which information should be delivered to the user interface.

With continuing reference to Figure 1, the architecture of the dashboard system 100 is described in more detail. Data is retrieved via the agents 140 from a number of different data sources 141. The data sources 141 can be presented in any number of different formats, including, without limitation, a client-specific data source 142, a pre-supported data source 144, a datamart 146, a web log analysis tool 148, and a third party application 150. A plurality of agents 140 are utilized, one for each of the data sources being incorporated into the dashboard system 100. Each of the agents 140 are specifically formatted to interact with its associated data source. In a preferred embodiment, the agents are computer software programs written specifically for each of the respective data sources.

The reaper 120 communicates with each of the agents 140 by a two way communication link 152 and is adapted to pull the data gathered by the agents on a scheduled basis. The reaper is adapted to utilize, for example, an XML push/pull routine in order to access and retrieve the data from the reaper 120. An agent polling configuration file 122 defines the necessary polling schedule information for the reaper 120. A datastore or data repository 124 serves as a data cache and stores the information gathered by the reaper 120. A repository manager 190, includes one or more memory caches 192, and further monitors and directs data that is transferred between the dashboard controller 180, the data repository 124, the reaper 120, and the alert detector 160.

An alert detector 160 is in two-way communication with the repository manager 190. The alert detector traps threshold conditions that are defined in an alert configuration file 162 and compares the threshold conditions to the data stored in the data repository 124 and processed by the repository manager 190. The alert detector 160 also sends notifications or alarms to a user based on definitions set in the alert

configuration file 162. Notifications and alarms can be in the form of an SNMP trap, email notification, pager notification, visual notification, or any other type of data transfer alert. For instance, if a data value stored in the data repository 124 is outside of a range defined in the alert configuration file, the alert detector will send an alarm or notification to the user indicating the "out of range" condition. As an example, if the dashboard system 100 is monitoring web-site hits per hour and the user wants to know if this value rises above a number that corresponds to the maximum capacity of the web server, the alert detector will recognize this condition and send an alert to the user.

The dashboard controller 180 is communicatively coupled to the repository manager 190 and generally functions to format the visualization of the data gathered by the agents 140 and passed through the reaper 120. A unique per-user configuration file 182 is customized for each user of the dashboard system 100 and defines for each user what information from the dashboard system 100 should be displayed to the user on a display device. The display device can be either a conventional PC monitor, a cell phone display, a PDA display (such as a device operating under the PALM or Windows CE operation systems), a pager display, or any other type of data display device. Additionally, information may be translated into a VoiceXML so that information may be accessed on a voice based system.

The dashboard controller 180 includes an HTML interface 188 that visually formats the data for display on a conventional web browser in the form of an HTML dashboard 184. The dashboard controller 180 also includes another interface 191 that formats the data for display on another type of dashboard such as one adapted to display or otherwise communicate WML, VoiceXML, or another data protocol. In this sense, it is contemplated that the dashboard controller 180 may include any number of interfaces. The dashboard controller 180 is preferably adapted to be expandable to incorporate interfaces for future types of dashboard viewing tools. Each of the components of the dashboard system architecture depicted in Figure 1 are preferably implemented in computer software residing on some form of fixed or otherwise permanent storage system such as a data server or other network of computers. However, the dashboard system 100 may also be implemented in a fixed hardware

format such as an Application Specific Integrated Circuit (ASIC) or a Field Programmable Gate Array (FPGA).

Figure 2 is another embodiment of a dashboard system 101 constructed in accordance with the present invention. The dashboard system 101 is similar in structure and function to the dashboard system 100 previously described except that a repository manager is not utilized. Instead, the reaper 120 is directly coupled to the data repository 124 which is in turn coupled to the dashboard controller 180.

Agent

Referring again to Figure 1, the dashboard system 100 collects data through its respective agents 140. Preferably, an agent is a program that fetches a set of information, formats the information into XML, or some other form of open coding language for storage in the data repository 124. Agents can be created using programming libraries developed from software languages such as PERL and JAVA as well as other programming languages. Each agent 140 can return one or more data items can be designated to return information in one of two ways. First, each of the agents 140 can return information dynamically. In this manner, the reaper 120 contacts the agent program via a Hyper Text Transfer Protocol (HTTP). The agent 140 then dynamically gathers the information and delivers XML to the reaper 120. Second, the agent can return information statically. In this manner, the agent is run at a time when the data is known to have changed, such as when a nightly processing routine is completed. The agent 140 then writes its XML out to a static file. The reaper 120 is further configured to access this static XML file. Agents can be formatted to get their associated data through a variety of means including SQL queries, text file processing, HTML scraping via an HTTP call, incoming email processing, SNMP, or spreadsheet exports. Other data gathering means are contemplated by the present invention and the above list is not meant to be limiting. Appendix A, which is hereby incorporated by reference, contains a listing of exemplary agents that have been developed for use in accordance with a dashboard system constructed in accordance with an embodiment of the present invention. Appendix A also includes selected information from future

agents that are contemplated by a dashboard system constructed in accordance with an embodiment of the present invention.

When an agent generates its data, it preferably converts that data into XML or another open code environment. The XML format is used for communication between the reaper 120 and the agents 140, and from the repository manager 190 to the dashboard controller 180.

As a typical example, dashboard XML files are structured in the following manner:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE dashboard SYSTEM
"http://www.xor.com/xml/dtds/dashboard.dtd">
<dashboard>
  ...individual data items...
</dashboard>
```

Dashboard Data Types - Generally, each piece of data assembled by the dashboard must conform to a data type. "Is a" notation is preferably used to declare inheritance between the different data types. When one data type "is a", it inherits the required and optional attributes from the given data type. The list compiled in Appendix B, which is hereby incorporated by reference, contains examples of data types supported by a dashboard system constructed in accordance with an embodiment of the present invention.

It is noted that the information in Appendices A and B only represents examples of agents and dashboard data types created for specific applications. It is contemplated that agents and data types may be created for any type of existing data or third party data processing application. Customized libraries can be established for use in creating agents. For example, PERL libraries may be established to supply methods for creating new dashboard objects, "freezing objects into XML", "thawing" objects into objects, and "selecting" a data item from within an XML structure. An example PERL library is included at Appendix C which is hereby incorporated by reference.

In addition to the agents described in Appendix A, the following are additional agent descriptions that may be created for a dashboard system constructed in accordance with an embodiment of the present invention. For example, the agents can be adapted to have the ability to "push" data to the reaper, rather than the data being polled. This may be accomplished by having the reaper supply an agent with a "callback" URL. The agent can then hit the callback URL to supply data to the reaper. An optional sequence/version number may be added to the dashboard XML item. Agents that write static HTML files would update the version number each time they write the file. Then, the reaper can use the version number to determine if the dataset has changed. Agents can also be adapted to have the ability to accept arguments from the reaper. For example, a generic "SQL Agent" could be created that accepts a SQL statement from the reaper and returns the results.

Reaper

The reaper gathers XML data by contacting the agents. The agents are contacted based on information supplied through a reaper configuration file (agent polling configuration file 122 from Figure 1). Preferably, the reaper configuration file supplies agent information for all agents, across all clients or there may exist multiple reaper configuration files; one for each client. The following represents a preferred example of a reaper configuration file:

Reaper Configuration File

Description	Indicates an individual agent that should be contacted to gather information.
Name	The name of each agent in the top level, or in a group (see below) must be unique. If an object disappears from the XML tree returned by an agent, it should be removed from the repository.
[illegible]	
[illegible]	The name of the XML file that the XML tree returned by this agent will be stored in.
[illegible]	The URL of the agent that supplies the XML data.
[illegible]	Indicates whether this agent is currently being refreshed

	(used). Valid values are **true** and **false**. Default: **true**
Path	The name of the directory that the XML file will be stored in. If not specified, will inherit from its group (in which case it *must* be set in the group; see below).
Refresh	The refresh frequency in seconds or a cron-style entry (see *man 5 crontab*). If not specified, will inherit from its group (in which case it *must* be set in the group; see below).
rate_type	Indicates that the reaper should automatically generate a rate field for this agent. The rate is calculated *after* new data is stored and the history is updated. If not specified, will inherit from its group (if set in the group). Valid values are **none**, **difference:*number***, **percent:*number***, **persecond:*number***, **median:*number***, **averagechange:*number***, and **averagevalue:*number***. Default: **none**.
Precision	Specifies a precision to limit all number elements to. A precision of "1" will make numbers look like "2.5". Default: leaves numbers as-is.
History	Indicates that the reaper should accumulate history for items within the XML tree returned by the agent. The history value specifies how many historical values should be kept. Default: **0**

The different rate_types that may be requested in the above configuration file example are further defined as follows:

none	Do not populate the rate field.
difference:number	Rate is calculated as: (current_value - history[*number*]). For example, difference:1 calculates the difference between the current and previous values. If *number* is greater than the size of the history, then the oldest item in the history will be used instead.
percentage:*number*	The rate_type is calculated in the same manner as difference, but a percentage change is placed in the rate field instead of the difference.
median:*number*	Rate is calculated as: (current_value + history[*number*])/2. For example, median:5 calculates the median between the current and 6th history value.
persecond:number	Rate is calculated as: (current_value - history[number])/ (current_timestamp - timestamp[number]). For example, persecond:1 calculates the difference between the current and previous in units per second.

average:number	Calculates the average of the history items, from current to history[number].
Averagechange :number	Calculates the average change, from current to history[number].

History Details - When scanning an XML tree returned by an agent, each element of type number (or subclass thereof) will have history accumulated for it by reading the previous list of values (if any) from the repository and adding the new value to the end of the list. Text and HTML objects will have their history stored by separating chunks of information by a null character. Other object types will simply have the latest value stored in the repository.

An element will not be pushed onto the history if its current timestamp is the same as the most recent item in the history list. This would occur when the agent program is setting the timestamp explicitly. For example, an agent may choose to set the timestamp to the time that the agent knows the data was last updated (with web logs, that is likely the timestamp of the previous day). If an agent provides its own history, then that history should be used instead, and not be overwritten.

An element of history length of X actually stores X+1 elements in its history. The "zero" element in the history array is the same as the current value/timestamp. Elements 1 to X+1 are the historical values/timestamps.

The following is an example of an Agent XML description:

```
<agent name="poll" refresh="20" path="/vitamins" rate_type="difference:0"
    source_url="http://www.vitamins.com/cgi-bin/dashboard/agents/poll"/>

<agent name="feedback" refresh="120" path="/vitamins" enabled='false' history='100'
    source_url="http://www.vitamins.com/cgi-bin/dashboard/agents/feedback"/>
```

Agents may be classified and categorized by groups. Each group preferably has the following characteristics associated with it.

Description	Specifies a group of related *agents*. Optionally specifies that the reaper should aggregate multiple items together into a single item.
Notes	The name of each group must be unique.
Attributes	
name	A unique name that can be used to refer to this group. Also used as the XML file name where the aggregated numbers will be stored if aggregation is enabled.
enabled	Indicates whether this group is currently being refreshed (used). Valid values are **true** and **false**. If **false**, then none of the *agents* in the group will be refreshed, independent of the settings of their individual flags. If **true**, then each *agent* can specify its own enabled status. Default: **true**
path	The directory location in which to store the resulting aggregated numbers, if aggregation is enabled (see below). If a path is defined here, it will be used as a default for *agents* in the group that don't have a *path* defined.
refresh	The refresh frequency in seconds or a cron-style entry (see *man 5 crontab*). If specified, it will be used as a default for *agents* in the group that don't have a *refresh* defined.
precision	Specifies a default precision to limit all number elements to in the group. A precision of "1" will make numbers look like "2.5". Default: leaves numbers as-is.
rate_type	Indicates that the reaper should automatically generate a rate field for the *agents* in this group. If specified, it will be used as a default for *agents* in the group that don't have a *rate_type* defined, and for aggregation, if it is enabled (see below). Default: **none**.
history	Indicates that the reaper should accumulate history for the aggregated items (see below), and how many historical values should be kept. If specified, it will also be used as a default for *agents* in the group that don't have a *history* defined. Default: **0**
Aggregate_ Numeric	Indicates that numeric aggregation should be performed on the values within this *group*. Valid values are **none, sum, average**, and **overlay**. Default: **none**
Aggregate_ Other	Indicates that aggregation of other, non-numeric data types should be performed on all the values within this *group*. Valid values are **none, merge**, and **overlay**. Default: **none**
Aggregate_ Maxoffset	Indicates the allowable time offset in seconds to use when aggregating history items. Default: **infinite**
Aggregate_ Rate_type	Indicates that a rate of change should be calculated for the aggregated values. See the "rates" section below for possible values. Default: **none**

Aggregation - When aggregation is enabled in a group, all of the child trees listed in the group will be scanned and a new tree will be constructed that contains elements from both trees. Corresponding elements in the tree (same name and location in the XML tree) are aggregated according to the following rules:

aggregate_numeric = sum|average
Items and their subtypes will be numerically summed/averaged. The latest value of each item will enter into the calculation of the aggregate value, regardless of its timestamp. History items will be summed/averaged if their timestamps fall within a given number of seconds of each other. If they do not, then the latest of the timestamped values is used.

aggregate_numeric = overlay
Overlays the new information on top of the old. For example, old_xml.overlay(new_xml) would overlay the contents of old_xml with the contents of new_xml. Any elements that are in new_xml that are not in old_xml will also be kept. Anything in the intersection of the two is defaulted to new_xml.

aggregate_numeric = none
No numeric aggregation is performed.

aggregate_other = overlay
Works the same as aggregate_numeric overlay, but with non-numeric items.

aggregate_other = merge
Appends as best possible non-numeric items. See specific data types below for definition.

aggregate_other = none
No non-numeric aggregation is performed.

Details for specific data-types - gauge items behave similarly to number items, with the added feature that the minimum value will be the greatest value of all the minimums of the items being aggregated, but it will be suppressed if not specified for any of the items. Likewise for the maximum (maximum value of all maximum values; suppressed if not specified for any item). "Text" items should have their contents concatenated, separated by newlines, in the order listed in the reaper configuration file. HTML items should have their contents concatenated, separated by an <HR>, in the order listed in the reaper configuration file. "Image" items should not be aggregated, and will generate an error if any are present in the XML trees specified within the group. "Chart" items should be aggregated as follows. The labels should be the union of all label values. The datasets should be the union of all datasets. Elements of

datasets with identical names should have the corresponding values of the datasets numerically summed/averaged/overlayed.

For the purposes of averaging, the actual number of items summed will be used to calculate the average. In other words, if there are 5 agents in the group, but one agent does not return a particular value, then the remaining 4 values will be summed and divided by 4 to calculate the average. If the agent wishes a "zero" value to be included in the average, it should return an item with an explicit value of 0.

An error will occur if items to be aggregated do not have exactly the same type (e.g. it will be an error to try to aggregate *text* and *image* or *number* and *gauge*). The individual XML trees retrieved from each agent will be stored as in the repository, so that a UI can retrieve the individual values as well as the aggregate. The historical values will be merged into a single list. Corresponding history items will be summed/averaged/overlayed (e.g.: sum = history_1[0] + history_2[0]). The reaper will do some level of error checking, as defined by *aggregate_maxtimeoffset*, to make sure that the timestamps are not different by more than this value. If they are, the history item with the newer timestamp is used, and a warning is logged.

The following is an example of a Group XML example:

```
<group name='stuff' enabled='false'>
  <agent name="poll" refresh="20" path="/vitamins"
      source_url="http://www.vitamins.com/cgi-bin/agents/poll"/>
  <agent name="feedback" refresh="20" path="/vitamins" enabled='false'
      source_url="http://www.vitamins.com/cgi-bin/agents/feedback"/>
</group>

<group name='all_web' path='/vitamins/web' aggregate_numeric='sum'
    aggregate_other='none' aggregate_rate_type='sum' aggregate_maxoffset='60'>
  <agent name="www1" refresh="60" path="/vitamins/web"
      source_url="http://www1.vitamins.com/cgi-bin/agents/web"/>
  <agent name="www2" refresh="60" path="/vitamins/web"
      source_url="http://www2.vitamins.com/dashboard/web"/>
</group>
```

This above example generates the following XML repository files:

/vitamins/web/www1.xml: the raw values from www1.vitamins.com
/vitamins/web/www2.xml: the raw values from www2.vitamins.com
/vitamins/web/all_web.xml: the aggregated values from both servers

When creating an aggregate object, the aggregate itself should be locked before it starts to process its group's items. This fixes possible deadlocking and agent data changing in the middle of an aggregation.

The following additional features of a dashboard reaper constructed in accordance with an embodiment of the present invention are also contemplated. To support an agent "push" enhancement, the reaper may supply the agent with a "callback" URL. The agent can then hit the callback URL to supply data to the reaper. To support an agent "version" enhancement, the reaper may be adapted to recognize the version number to determine if the dataset has changed. To support an agent "argument" enhancement, the reaper may be adapted to support attribute(s) to send arguments to an agent that is designed to be "generic." Exceptions may be indicated if the reaper is not able to contact an "enabled" agent. For example, an email may be sent to an administrator.

The reaper may also be adapted so that the reaper children that are performing a request can "time out." This timeout value would default to something reasonable (i.e. 60 seconds) and may be overwritten by an agent/group specific "timeout" attribute. The controller can be adapted to make a dynamic request for information. If the reaper does not have the request in the datastore, the reaper would then contact the appropriate agent for the information.

A configuration file for each client may be set up and a configuration file for "generic" information (e.g.: News, Stock Market Quotes) may also be set up. The reaper should be able to be told to re-read a specified configuration file.

Alert Detector

An "alert" can be set on any Dashboard indicator and is triggered when an indicator does, for example, one of the following: 1) descends below a certain threshold; 2) rises above a certain threshold; or 3) is on or between two numbers. When an alert is defined, a severity and an action are associated with it. For example, severity levels of zero through three can be defined (this can be expanded upwards if needed). Actions that can be taken include: 1) Visual: An element in the indicator changes, such as an arrow turning red; 2) Email: An email is sent to a specified address. This address could be a normal user's address, a list of users or a pager email address (most alpha-numeric pagers have associated email addresses); 3) SNMP: An SNMP "trap" can be triggered. SNMP stands for "Simple Network Management Protocol" and is used by network monitoring software to gather information and alerts about routers, switches, machines and other network-enabled devices. For example, SNMP could be used to alert a user's monitoring software. In order for the controller to know if an alert has been triggered, the Alert Detector writes out an XML file that contains the appropriate alert information.

Preferably, an alert configuration file supplies alert information for all agents, across all clients or there may exist an alert configuration file for each user or client. The following describes the contents of the configuration file that are wrapped with an alerts XML tag.

Action

Description	Indicates an action to take when an alarm is triggered.
Notes	The name of each *action* must be unique.
Attributes	
name	The name of the action. This is used in the *monitor* tags to indicate which action to take when an alert is triggered.
Type	The type of action to take. Can be one of: • display: Indicate that an alert has happened on the dashboard display • email: Email a notification to someone that an alert has occurred

	• snmp: Trigger an SNMP trap
destination	The contents of the destination depends on the action *type*. • display: The alert file to write its alert XML to. • email: Email address(es) of the people to notify. Note that this can also be a pager email address • snmp: MIB
color	Only set for actions of type "display". The color is made available to the templating engine.
template	Only valid for actions of type "email". Email template to use to compose the email.
subject	Only set for actions of type "email". The subject of the email being sent.
once	Only set for actions of type "email". Indicates if an alert should be sent only once (set to "yes"), or every time that the alert detector uses this action type.
snmp	Only set for actions of type "snmp". The MIB address to send an the alert to.

XML	<action name="EmailAdmin" type="email" template="emailbody" destination="admin@xor.com" subject="Alert" once="yes" /> <action name="Down" type="display" color="#ff0000" destination="ALERTS" /> <action name="Neutral" type="display" destination="ALERTS" color="#77ccff" /> <action name="Up" type="display" destination="ALERTS" color="#66ffcc" />

Monitor

Description	Surrounds a block of *level* tags and specifies the XML to look at to detect an alert.
Notes	The name of each *monitor* must be unique.
Attributes	
name	The name of the monitor.
Source	The name of the source XML file or an alias to the source XML file (specified in data.xml).
select	A selection of a data item inside the source. Only required if the *source* doesn't point to the item to look at.

XML	<monitor name="DJIA" source="DJIA" select="difference/value"> ... </monitor>

Level

Description	Defines the actual alert ceiling/floors for this monitor. A *level* may only be specified inside a *monitor*.
Notes	The name of each *level* within its *monitor* must be unique.
Attributes	
name	The name of the action. This is used in the *monitor* tags to indicate which action to take when an alert is triggered.
Eff_type	Effective start date, specific as *YYYYMMDD*. If one is not specified, it will begin immediately.
Low	If both a *low* and *high* are set, the *low* specifies the floor for this alert. If an alert falls on or between the *low* and *high* (inclusive), this alert is triggered. If only a *low* is set, then an alert is triggered if the value falls on or below the *low*.
High	If both a *low* and *high* are set, the *high* specifies the ceiling for this alert. If an alert falls on or between the *low* and *high* (inclusive), this alert is triggered. If only a *high* is set, then an alert is triggered if the value falls on or above the *high*.
XML	<level name="down" eff_date="20010101" low="-25" high="0"> ... </level>

Result

Description	Defines tags to set in the template engine for this level. A *result* can only be specified inside a *level*.
Attributes	
directional	Tag used in the templating engine to indicate part of the name of a directional graphic to display.
Msg	Message to display to the end-user when they place their mouse over the alert icon.
Severity	The severity level for this alert. Ranges can be 0 through 3. 0 is typically a visual-only alert (change a directional graphic red, for example), while alerts 1 through 3 typically indicate a more severe problem. An alert icon is displayed to the user for severity levels 1 through 3.

XML	<level name="down" eff_date="20010101" low="-25" high="0"> <result directional="red" msg="Falling" notes="" severity="0"/> </level>

Log

Description	Defines the *action* that this alert uses.
Attributes	
action	Indicates the *action* name to trigger for this alert.
destination	Destination XML file; overwrites the *destination* specified in the *action*.
XML	<level name="down" eff_date="20010101" low="-25" high="0"> <log action="Down" destination="DJIAalert"/> </level>

The following is an example of an XML scheme for a component that represents a DJIA stock market index alert.

```
<alerts>
 <action name="Down" type="display"
          color="#ff0000"
          destination="ALERTS" />
  <monitor name="DJIA" source="DJIA" select="difference/value" >
    <level name="down" eff_date="20010101" low="-25" high="0">
      <result directional="red" msg="Falling" notes="" severity="0"/>
      <log action="Down" destination="DJIAalert"/>
    </level>
    <level name="down25" eff_date="20010101" low="-50" high="-25">
      <result directional="red" msg="Down over 25" notes=""
severity="1"/>
      <log action="Down" destination="DJIAalert"/>
    </level>
    <level name="down50" eff_date="20010101" low="-75" high="-50">
      <result directional="red" msg="Down over 50" notes=""
severity="2"/>
      <log action="Down" destination="DJIAalert"/>
    </level>
    <level name="down75" eff_date="20010101" low="-75">
      <result directional="red" msg="Down over 75" notes=""
severity="3"/>
      <log action="Down" destination="DJIAalert"/>
    </level>
```

```
    </monitor>
</alerts>
```

The above XML alert is set up to behave in the following manner. If the change in the index since the market opened is positive, no alert is set. If the change in the index since the market opened is between 0 and –25, alert severity 0 is triggered, and the down-arrow is made red. If the change in the index since the market opened is between -25 and –50, alert severity 1 is set, the down-arrow is made red and an "Alert Severity 1" icon is displayed on the indicator. If the user places their mouse over the indicator they will see the text "Down over 25". Alert severity 2 is set when the change in the index since the market opened is between -50 and –75. The down-arrow is made red and an "Alert Severity 2" icon is displayed on the indicator. If the user places their mouse over the indicator they will see the text "Down over 50". If the change in the index since the market opened is over -75, alert severity 3 is set, the down-arrow is made red, and an "Alert Severity 3" icon is displayed on the indicator. If the user places their mouse over the indicator they will see the text "Down over 75".

It is also contemplated that in a dashboard system constructed in accordance with an embodiment of the present invention, the alert detector may have the following additional features. Users can have the ability to specify Key Performance Indicators. The controller gets its information from a user/client-specific output file and there are user/client-specific alarm configuration files. A single alarm file may "include" another. Duplicate alarms would be overwritten by the alarms in the included file. Alarms may activate/deactivate based on a start and end date and time (i.e.: today the ceiling is 10, tomorrow the ceiling is 20). Alarms may be dynamically set by requesting ceiling/floor information from another data source. A user interface may be created for changing alarm information and alerts may be set that span different agents.

Dashboard Controller

The dashboard controller configuration is composed of a large set of files that can be configured per user. These include HTML/ASCII templates, data location

definitions, and the display configuration. The basic configuration is composed of the following directory hierarchy:

```
dashboard
|
+-conf
| |
| +-full Default configuration directory; contains a configuration file for
each user
|
+-templates
  |
  +-default
  | |
  | +-framed
  | +-htmltable
  | +-<other types of rendering templates>
  |
  +-<username> Optional; this is only used if the user/client needs a custom
template
    |
    +-framed
    +-htmltable
    +-<other types of rendering templates>
```

The *conf* directory contains the XML files that define the screen display. Each user has one configuration file preferably named *username.xml*. In addition, there is a file that defines aliases that reference the dashboard data repositories. It is preferably called *data.xml*, but may be changed so that the filename is specified in the *username.xml* file.

The *templates* directory contains a *default* directory, and may contain a directory hierarchy for each user. The *default* directory is used if a template file *can't* be found in the user directories. These directories contain templates, ending in *.tmpl*, specific for the display renderers. Each type of renderer contains a set of templates that correspond to either the type of dashboard object or a template type specified in the user configuration. The templates are processed to replace a set of tags with the data values from the data sources. There are five tags that are used to do this:

<TMPL_VAR NAME="*name*">
This is a basic "name" substitution. If a data element specified by *name* exists in the user configuration file, then that value will be inserted into the output.

<TMPL_IF NAME="*name*"> ... </TMPL_IF>
This is a conditional statement. The text between the beginning and ending TMPL_IF tags will be included in the output if the value of *name* evaluates to true in the Perl sense: either non zero or a string. This tag also has a <TMPL_ELSE> tag that can be used for if-then-else scenarios.

<TMPL_UNLESS NAME="*name*"> ... </TMPL_UNLESS>
This is the reverse of the TMPL_IF tag. It outputs its text if the value of *name* evaluates to false in the Perl sense: 0 or undefined.

<TMPL_LOOP NAME="*name*"> ... </TMPL_LOOP>
This is the basic looping mechanism. The TMPL_VAR and TMPL_IF tags can be contained within the loop. The loop is ran until it has iterated over all of the data values contained in the *name* (as specified in the user configuration file).

<TMPL_INCLUDE NAME="*filename.tmpl*">
Includes the named file in this template. Works just like a server-side include.

The following is a preferred example of a template:

```
<TABLE>
<TR>
<TD>name = <TMPL_VAR NAME="name"></TD>
<TD>Description = <TMPL_VAR NAME="description"></TD>
</TR>
<TMPL_LOOP NAME="load">
<TR>
<TD>
<TMPL_IF NAME="machine_name">
<TMPL_VAR NAME="machine_name">
</TMPL_IF>
</TD>
<TD>
<TMPL_IF NAME="machine_load">
<TMPL_VAR NAME="machine_load">
</TMPL_IF>
</TD>
</TR>
```

```
</TMPL_LOOP>
</TABLE>
```

The following is a preferred example of an output file:

```
<TABLE><TR>
<TD>name = Load Times</TD>
<TD>Description = Load Times on Client Machines</TD>
</TR>
<TR>
<TD>
Machine 1:
</TD>
<TD>
1.5
</TD>
</TR>
</TR>
<TR>
<TD>
Machine 2:
</TD>
<TD>
3.2
</TD>
</TR>
</TABLE>
```

Data configuration files contain XML entities that define aliases (shortcuts) to the Dashboard data repository items. These aliases are used in the controller configuration files. The default data file is preferably located in a file labeled /info/dashboard/ conf/full/data.xml, but there may be one file per client/user (named the client/user). The aliases add an abstraction layer that may be able to be used to reduce the overall number of templates that are needed.

The following is a preferred example of a data configuration file:

Load

Description	Associates an alias name to a data file
Notes	At a minimum, a *load* should be created for each XML data file that will be accessed in the user's display configuration.

Attributes	
Name	The name of the alias. The name of each *load* should be unique.
File	The full filename of the repository data file. In a future release, this should be changed to be relative to the base directory of the dashboard repository.
select	A path into the specified *file*'s XML data tree. This attribute allows an alias to point to a data node that is deep within the XML tree. If a *select* is not specified, then the alias simply points to the data file.
XML	<load name="servicemetrics" file="/info/dashboard/repository/servicemetrics/sm.xml"/> <load name="historical_stats" file="/info/dashboard/repository/tsn/weblog/historical.xml" select="sitestats"/> <load name="weblog" file="/info/dashboard/repository/tsn/weblog/weblog.xml"/> <load name="weblog_box1" file="/info/dashboard/repository/tsn/weblog/box1.xml" select="sitestats/www.sportingnews.com/today"/>

Each display configuration file defines the display of a specific user or client. They work with the aliases defined in the data configuration. The configuration file contains the following hierarchy of elements which are described in more detail in Appendix D, which is hereby incorporated by reference:

- navorg
    - section
        - construct
            - item

All of the tags can contain one or more *attr* elements, which behave similar to an entity/attribute that is included within the main element tag. This is primarily used for readability, when an item has a lot of attributes. For example, a layout attribute can be specified with either:

```
<section position="1"> ... </section>
(or)
<section>
 <attr name="position">1</attr>
 ...
</section>
```

There are a number of variables that are automatically populated for templates. Whether or not a variable is populated depends on the data elements that a template is given and the item *types* being displayed. A list of variables are found in Appendix E, which is hereby incorporated by reference.

The templates used by the dashboard controller are accessed by using template="*templatename*" in a *construct*. A number of templates have already been developed and are listed in Appendix F, which is hereby incorporated by reference. These have been developed to provide a high level of flexibility across many different types of metric measurements.

A dashboard controller utilized in a dashboard system constructed in accordance with an embodiment of the present invention may also be adapted to include the following features. Object caching may be added in order to speed up the display. "Include" files may be added inside the Controller configuration files, so that similar per-user configuration files can be easily made. Included files would overwrite duplicate elements. Display thresholds for KPIs (Key Performance Indicators) and alerts may be graphically displayed and alternate graphing engines may be used.

IMPLEMENTATION EXAMPLE OF VISIONPORT DASHBOARD

The following description provides a preferred embodiment of a dashboard system constructed in accordance with the present invention. The example is representative of an implemented Dashboard system as designed and built by XOR, Inc. of Boulder, Colorado. The dashboard is preferably password protected and can be

customized for different executives, departments or partners of an eBusiness. Although the embodiment described herein is presented in the form of a HTML internet-based environment, a Wireless Application Protocol (WAP) version of the Dashboard is also contemplated in order to provide dashboard information to a number of different handheld or wireless devices operating on a variety of platforms.

Figure 3 represents an exemplary main dashboard screen 300 of a dashboard system constructed in accordance with an embodiment of the present invention. Preferably, the dashboard screen 300 is broken down into a grid consisting of up to sixteen indicators although more or less indicators may be utilized. As shown in Figure 3, the sixteen indicators are labeled (and described) as 305 (control panel), 310 (Dow Jones Stock Market Indicator), 315 (NASDAQ stock market ticker), 320 (S&P 500 stock market ticker), 325 (web site statistics), 330 (non-web site statistics), 335 (advertising statistics), 340 (revenue statistics), 345 (previous day web site statistics-yesterday), 350 (previous day web site statistics-week ago), 355 (average web site statistics), 360 (1 month average web site statistics), 365 (home page load time statistics), 370 (average system load time statistics), 375 (web page visits per day), and 380 (registered users). Preferably, a blank box is displayed whenever there are fewer than 16 indicators in use (e.g. when a 3x3 or 3x4 array of indicators are used). Appendix G, which is hereby incorporated by reference, provides a table that describes of each of the above indicators in further detail. Upon opening in a browser-based environment, the dashboard preferably will resize the browser window to the dimensions necessary to display the indicators.

The uppermost left indicator 305 is referred to as a control panel and is preferably present on all dashboard embodiments. Figure 4 shows the control panel 305 in further detail. The control panel 305 contains a menu of items that includes links to various sub-components of the dashboard system. For example, link 402 accesses the "Analysis Center". The analysis center generates custom reports as defined by a particular user. Preferably, the dashboard system is adapted to link to a third party reporting tool, such as the Brio Reporting engine in order to generate these

custom reports (See http://www.brio.com, the details of which are hereby incorporated by reference).

A Link 404 accesses the Site Log Reports which is preferably adapted to link to a third party web log analysis package such as WebTrends. The dashboard may incorporate multiple "Log Reports" links, for example, when a client has more than one web-site, or more than one web-log analysis profile for their site. A Link 406 shrinks the Dashboard to a smaller version that is better adapted for smaller displays and more conducive to put in a corner of a display where it can remain visible while a user engages in other activities. An Exit link 408 closes the dashboard browser window.

Other links can also be incorporated into the control panel 305. For example, various links to other 3rd party applications that a user needs to access regularly or links to other company information may be incorporated into the control panel.

Each of the indicators detail a different aspect of a user's eBusiness. There are several generic types of indicators, which are detailed below. Indicators with other formats, can be created if desired. As an example, Figures 5 - 7 detail the indicators 310, 325, and 365 respectively and illustrate the details of each. A title bar 420 is positioned at the top of each of the indicators and includes an indicator title 422, a detail icon 424, a report icon 426, and a help icon 428. The Indicator Title 422 shows the logical title of the indicator being displayed. Since the displayed title may be truncated or otherwise abbreviated, a user may utilize the help icon 428 to get a full detailed description of what information the indicator is displaying. The detail icon 424 (Magnifying Glass) retrieves a more in-depth display of the information. This may take the form of another grid of indicators e.g., "Today's Stats" indicator, or of a popup window e.g., "HP Load Time" indicator. Details on these popup windows are provided below. The report icon 426 (Paper) links to a more in-depth report of the indicator. In many instances this will link to a specific report in the Analysis Center 402. However, it can also link to other URLs or packages. For example, the DJIA report icon preferably links to the Yahoo Financial site's DJIA details page. The Help Icon 428

(Question-mark) links to a context sensitive help dialog window. The help text provides context on the metric, e.g., its definition, how it's calculated, etc.

A Primary Indicator Number Bar 440 includes a primary indicator value 442, a unit field 444, a rate of change field 446, a directional indicator 448, a recent change field 450, and an alert icon 452. The primary indicator value 442 reflects the most current information the indicator has. The unit field 444 displays the units of the primary indicator value, such as "pv" which stands for "page views" or "sec" which stands for "seconds." The Rate of Change field 446 indicates the rate of change between the current and the last data point retrieved. For example, the "Today's Stats" indicator in Figure 6 has a rate of change of 27.7 page views per second. This means the client's site was averaging 27.7 page views per second over the last 10 minutes. The directional indicator 448 indicates the current direction of data displayed in the indicator. The Recent change field 450 is the amount that the indicator has changed between the current and last data point. For example, the "HP Load Time" in Figure 7 has gone up 2.48 seconds in the last hour. The recent change field 450 may sometimes reflects a different time period. For example, with the stock market indicators, the number illustrates the change in the indicator since the start of the trading day. The alert icon 452 indicates if a data item becomes extremely high or low. The alert thresholds are configured on a per-indicator basis. Preferably, alerts can have four levels, from 0 to 3. A level 0 indicator typically does not display an alert icon, but will instead turn the up/down arrow red. Levels 1 through 3 display an icon. If you place your mouse over an alert icon, text will appear and detail the alert.

A Data Refresh Information bar 470 includes a date field 472 and a refresh interval field 474. The date field 472 indicates the date and time that the currently displayed information represents. The refresh interval field 474 describes how frequently the information in this indicator changes. Preferably, the refresh interval comes in three forms: minutes (e.g. 10m), hours (e.g. 6h), or days (e.g. 1d). This is not, however, the amount of time that passes between refreshing the information on the screen. For example, a refresh interval of "1d" indicates that the information is updated

once per day. However, the dashboard system 100 may check for new information every hour so the user receives refreshed information soon after it becomes available.

An Additional Information bar 480 includes either a Graph field 482 or a table data field 484. The graph field 482 graphs the history of the current indicator or the history of information relevant to the current indicator. Graphs may be displayed as lines, bars, or pie charts. The table data field 484 displays either a breakdown of items that make up the main indicator number or additional, less critical statistics that relate to the indicator. For example, a "Registered Users" indicator may detail the different types of users and the number of each.

By selecting the detail icon 424 a detail pop-up window will be displayed. Figures 8 - 10 show representative examples of detail pop-up windows 500, 600, and 700 respectively. Each of the detail popup windows displays a more in-depth view of the data provided by the associated indicator. The detail popup windows may come in a variety of formats including, for example, graphs (Figure 8) and tables (Figures 9 and 10). Figures 8 and 9 are an example of a line graph and its associated table. Figure 10 is an example of a detailed table that does not include an associated graph.

Referring to Figure 8, the detail pop-up window 500 includes a primary indicator number 502, a data refresh field 504, a high/low field 506, a graph/table toggle field 508, and a graph field 510. The primary indicator number 502 has the same elements that its associated summary Dashboard indicator does, including units, rate of change, up/down arrow, number under arrow, and alert icon. The Data Refresh field 504 also displays the same information as on its associated summary Dashboard indicator. The High/Low field 506 displays the high and low of the primary indicator number during the time-period in the graph. The Graph/Table Toggle field 508 allows a user to toggle between the graph and table versions of the same data. This field will not appear if only table data is available. The Graph Data field 510 displays the history of the indicator number in either line, bar or pie chart format. Preferably, multiple items can be graphed. For example, on the "HP Load Time" graph in Figure 8, load times for the home page both with and without ads and a standard "Internet Index" are

graphed. Data for a point on the graph will be displayed on mouse-over. Figures 9 and 10 include a Table Data field 512 that displays the history of the indicator number and possibly the history of related statistics as well in a table form rather than a graph form.

In addition to the ability to retrieve proprietary information from a legacy system or another internal corporate database, a dashboard system constructed in accordance with an embodiment of the present invention is preferably adapted to interact with and retrieve data and other business metrics from third party software and web-based applications. For example, the VisionPort Dashboard has the ability to extract and display information generated from the third party application CommerceTrends. In the VisionPort dashboard system, nearly every piece of information that is visible in an original CommerceTrends report can be displayed as a VisionPort Dashboard Indicator. For example, the following table details some of the standard CommerceTrends reports that may be extracted and displayed by the VisionPort dashboard system:

Activity	
	Summary for Report Period
	Month Visits/Hits
	Week Visits/Hits
	Day of the Week Visits/Hits
	Hour of the Day Visits/Hits
	Number of Pages Viewed per Visitor
	Length of Visit by Visits/Views
	By Kbytes Transferred Over Time of Day
Advertising	
	Views and Clicks (summary)
	Advertising Views (visits/views over time)
	Advertising Clicks (visits/views over time)
	Browsers and Systems
	Top Browsers
	Microsoft Explorer Browsers
	Netscape Browsers
	Top Platforms
	Errors
	Technical Statistics
	Dynamic Pages & Form Errors
	Client Errors
	Page Not Found Errors

	Server Errors
Files	
	Top Entry Files
	Least Requested Entry Files
	Hits Over Time
	Most Downloaded Files
	Most Uploaded Files
	Top Directories
	Most Downloaded File Types
Hits Over Time	
Marketing Campaigns Summary	
	Revenue forecast by Marketing Campaign
	Revenue forecast by Marketing Campaign and Product
	ROI by Marketing Campaign
	ROI Percent by Marketing Campaign
Pages	
	Top Entry Pages totals and over time
	Least requested Entry Pages totals and over time
	Top Exit Pages totals and over time
	Top Content Groups
	Single Access Pages
	Dynamic Pages and Forms
	Views Over Time
Parameter Analysis by Visits/Hits	
Paths	
	Top Paths Through Site
	Top Destination Paths Through Site
Product Summary	
	Revenue Forecast by Product and Visitor Type (qualified/non-qualified)
	Revenue Forecast by Product and Referrer
Referrers	
	Top Referring Sites
	Top Referring URLs
	Top Search Engines
	Top Search Phrases
	Top Search Keywords
Server Cluster Load	

Balance	
	By Kbytes Transferred
	By Hits
Visitors	
	New vs. Returning Visitors (Totals and Over Time)
	Top Authenticated Visitors (Visits/Hits with Over Time)
	Top Visitors (Visits/Hits with over time)
	Visits Over Time
	Number of Visits

Figures 11 and 12 represent an example of a dashboard setup in accordance with an embodiment of the present invention where a "global" association has one dashboard configuration (Figure 11) and each of any number of association franchises has a second dashboard configuration (Figure 12). In the association dashboard, figures are compiled for the association as a whole rather than for any one particular franchise, where the franchise dashboard is particularized for one particular outlet, store, or location. Information that is not of particular importance to a franchise is not displayed on the franchise dashboard. The same is true for the association dashboard. Individual dashboards may be set up for each individual franchise in an association.

Although the present invention has been described and illustrated in the above description and drawings, it is understood that this description is by example only and that numerous changes and modifications can be made by those skilled in the art without departing from the true spirit and scope of the invention. The invention, therefore, is not to be restricted, except by the following claims and their equivalents.

Appendix A

Agent Descriptions

ATG Dynamo

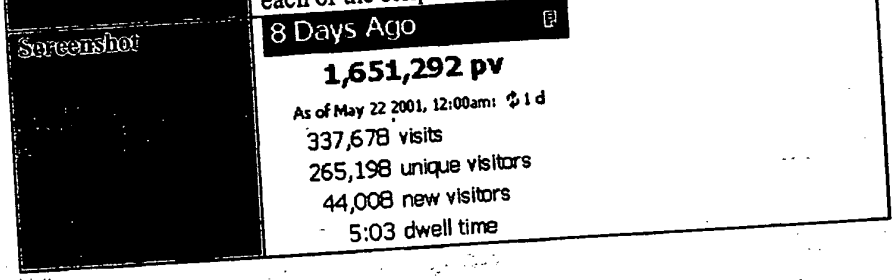
Language	Perl
Version	1.0
Reusability Level	High
Customers Using	UAP
Typically Updated	10 minutes
Collection Method	SNMP
Statistics Returned	d3System • sysServerName • sysStatus • sysUpTime • sysTotalMem • sysFreeMem • sysNumInfoMsgs • sysNumWarningMsgs • sysNumErrorMsgs d3LoadManagement • lmIsManager • lmManagerIndex • lmIsPrimaryManager • lmServicingCMs d3SessionTracking • stCreatedSessionCnt • stValidSessionCnt • stRestoredSessionCnt • stDictionaryServerStatus d3DRPServer • drpTotalReqsServed • drpTotalReqTime • drpAvgReqTime • drpNewSessions d3DBConnPooling • dbPoolID • dbMinConn • dbMaxConn • dbMaxFreeConn

	• dbBlocking • dbConnOut • dbFreeResources • dbTotalResources
Bugs	None known
Implementation Details	Collects various stats from the Dynamo server via SNMP. Configured through the "dynagent.cfg" file. This script could easily be used to create other SNMP agent; the configuration file style makes it very versatility.

CommerceTrends

Language	Perl
Version	1.0
Reusability Level	High
Customers Using	TSN
Typically Updated	Daily
Collection Method	Parsing text file output from CommerceTrends
Statistics Returned	Daily Stats: • Page Views • Page Views % of Month • Dwell Time • First-time Visits • Unique Visitors • Visit Minutes • Visits Monthly Stats: • Page Views • Visits • Unique Users • Users Visited Multi • Users Visited Once • Return Visits • First-time Visits • No-cookie Visits
Bugs	None known
Implementation Details	CommerceTrends uses Perl scripts to generate all of its reports. /usr/local/commercetrends/wt_script/report.pl was modified to output statistics to a flat text file as the reports were generated. The text files live at: vp3:/u05/commercetrends/xordata/incoming.

Appendix - 2

	CommerceTrends is configured to run the script /info/dashboard/bin/ct_postprocess *sitenumber* when it finishes processing the logs. This script adds the information in the text file to a DBM file for the specified site. Another script, ct_agent, is called to generate the XML file that's picked up by the reaper. Nearly any statistic that shows up in a report can be imported into VisionPort. It just takes some changes to each of the scripts mentioned above.
Screenshot	

Appendix - 3

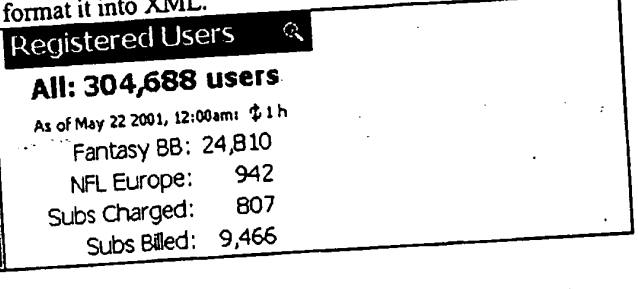
Foglight

Language	Perl
Version	1.0
Reusability Level	High
Customers Using	XOR
Typically Updated	15 minutes
Collection Method	Text Scraping
Statistics Returned	Machine Load and CPU usage
Bugs	None known
Implementation Details	Takes the Foglight site identifier in on the PATH_INFO. For example: NT_system@web-nt8.xor.com. Based on the type of system (determined from looking at the site identifier for "NT", "vision2" (bsdi), or "Sun", it executes a foglight command that outputs stats for that system for the past hour. Other statistics can be gathered. See the script for pointers to the foglight commands used to determine the statistics that are available.

HTTP/Text Scraping Example

Language	Java
Version	1.0
Reusability Level	High
Customers Using	None
Typically Updated	N/A
Collection Method	HTTP/Text scraping
Statistics Returned	Stock ticker
Bugs	None known
Implementation Details	Java version of the Perl-based quote fetcher for Yahoo stock ticker.

Informix DB Data Extraction

Language	Perl
Version	1.0
Reusability Level	Low; TSN-specific
Customers Using	TSN
Typically Updated	10 minutes
Collection Method	SQL Queries
Statistics Returned	Registration information from the TSN user database
Bugs	None known
Implementation Details	TSN has lots of user data stored in an Informix database that lives on tsn-db1. Scripts on tsn-db2 connect to the DB, extract relevant information with SQL queries, and format it into XML.
Screenshot	

PingTier1

Language	Perl
Version	1.0
Reusability Level	XOR-specific
Customers Using	MCC
Typically Updated	1 day
Collection Method	Text file scraping
Statistics Returned	Average and Maximum ping time to XOR's tier-1 providers over the past 31 days.
Bugs	None known
Implementation Details	Some shell scripts ping each of XOR's tier-1 provider connections every 5 minutes and store the results for 31 days. The agent looks at the files and averages the maximum and average ping times.

RealMedia OpenAdstream

Language	Perl
Version	1.0
Reusability Level	Low, unless another client uses OpenAdstream
Customers Using	TSN
Typically Updated	10 minutes

Appendix - 5

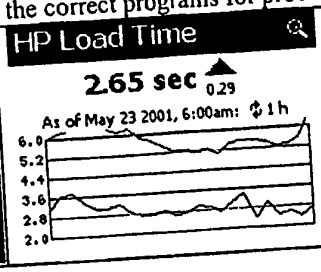
Collection Method	Log parsing
Statistics Returned	Number of ads served per ad position.
Bugs	None known
Implementation Details	Constantly reads the OAS log files on each machine and counts the number of as served. The agent reads a DBM file (the same one as the near-real time log parser.
Screenshot	

Remedy

Language	Perl
Version	1.0
Reusability Level	High
Customers Using	XOR
Typically Updated	Hourly
Collection Method	SQL Queries against Remedy database
Statistics Returned	Requests, Pending, Resolved, Min/Max/Avg TTR for each priority. Service Communications (email/phone). List of all incidents Started, Resolved, or Pending in a given time period.
Bugs	The Service Communications piece may not be accurate, as it's unclear how tickets that are opened automatically from HPOV are logged as "communicating".
Implementation Details	Connects via the following: 'dsn' => 'dbi:Oracle:host=evolve.xor.com;sid=ARSPR1', 'username' => 'vision_report', 'password' => 'vision_report'

ServiceMetrics

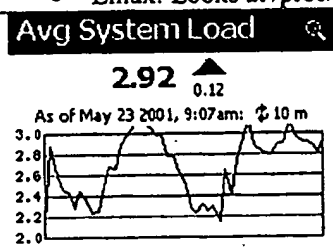
Language	Perl
Version	1.0
Reusability Level	High
Customers Using	TSN
Typically Updated	Hourly
Collection Method	Incoming email parsed
Statistics Returned	Page load time history for the past 24 hours Service Metrics' Internet Index and eCommerce Index

Bugs	Sometimes one data item has an additional field (data that it has collected so far this hour). If this data field exists, it should be deleted.
Implementation Details	ServiceMetrics was configured to email a report to servicemetrics@xor.com. This email address pipes the output to a script that sends the data over to dev-linux for processing. On dev-linux, a script run from cron every few minutes looks for incoming files and delivers them to the correct programs for processing.
Screenshot	

SQL Query Skeleton

Language	Java
Version	1.1
Reusability Level	High
Customers Using	None
Typically Updated	N/A
Collection Method	SQL Queries
Statistics Returned	Columns/rows requested through query
Bugs	None known
Implementation Details	First-pass Java SQL query agent. It returns generic information, based on the SQL query specified to the agent. Arguments: • URL: string used to access the database • username: database username • password: database user password • labeledColumns: Boolean: whether or not to label columns in the XML output. • labeledNumeric: Boolean: if the labels should be numeric or strings; if strings, they're the SQL column names. • labeledZeroBased: Boolean: if the labels are 0- or 1-based. Even if labeledNumeric is false, this variable is still used to determine how the returned "label" item's names are output as well as the returned rows' names. • query: SQL query to execute

System Statistics

Language	Perl
Version	1.0
Reusability Level	High, but only on same operating system. Currently implemented on BSDI and Linux. Easy to port to another O.S.
Customers Using	TSN
Typically Updated	10 minutes
Collection Method	System Calls
Statistics Returned	1, 5, and 15 minute load time Uptime of the machine in seconds ("elapsed" data type) Approximate machine CPU utilization
Bugs	None known
Implementation Details	Uses a system call to "uptime" to gather 1, 5, and 15 minute load times and the uptime of the machine. To gather information about CPU utilization, it uses: • BSDI: /usr/sbin/iostat -c 2 10 • Linux: Looks at /proc/stat in 10 second intervals
Screenshot	

Web Log Statistics (XOR): Near-realtime

Language	Perl
Version	1.0
Reusability Level	High on systems that use standard web logging. Would need to be modified to be used on NT.
Customers Using	TSN
Typically Updated	10 minutes
Collection Method	Parsing web logs on-the-fly
Statistics Returned	For today since midnight: Hits, page views, visits, page views per visit, page views per content area
Bugs	None known
Implementation Details	A daemon runs on individual machines and parses the web log files on-the-fly. Every 10 minutes the daemon outputs the current statistics to a DBM file (/info/httpd/statsum/*YYMM*/*sitename*/dashboard). The agent reads this DBM file to generate its XML.

Screenshot	Today's Stats **496,601 pv** (30.5/s) As of May 23 2001, 9:07am: 10 m 4,384,060 hits (304.2/s) 134,745 visits (9.0/s) 10,530 reg visits 54 new reg

Web Log Statistics (XOR): Historical

Language	Perl
Version	1.0
Reusability Level	High
Customers Using	TSN
Typically Updated	Daily
Collection Method	Extract statistics from DBM file generated from XOR log parser
Statistics Returned	Hits, page views, visits, page views per visit, page views per content area for: • Yesterday • 8 days ago • 7 day average • 28 day average A 28 day running history is also available for "visitors".
Bugs	None known
Implementation Details	A program on the log parsing machines (web-log and tsn-logs) analyzes log files nightly. When it's done with a day, it outputs stats for the day into a DBM file (/info/httpd/statsum/*YYMM*/*sitename*/misc). The agent reads this DBM file to generate its XML.

Web Log Statistics (XOR): Historical – Last Month Only

Language	Perl
Version	1.0
Reusability Level	High
Customers Using	MCC
Typically Updated	Daily
Collection Method	Extracts statistics from past monthly reports by text scraping.
Statistics Returned	Hits, page views, visits, page views per visitor, megabytes transferred area for, this month, last month, and a year ago last month.
Bugs	None known

Implementation Details	Scrapes files on web-log.xor.com from /info/logs/old-logs/*YYYY*/*Mon*/www/*sitename*. Is able to deal with web log reports that have been gzipped as well. The script gets the sitename from the PATH_INFO environment variable. It's called like: http://web-log.xor.com/cgi-bin/agents/weblog_stats_monthly_history/www.sitename.com

WebPosition Gold

Language	Perl
Version	1.0
Reusability Level	High
Customers Using	None
Typically Updated	Weekly
Collection Method	HTTP calls, scraping the returned tab-delimited text
Statistics Returned	Number of keywords searched for, total number of matches, average position, and for each search engine: • Number of matches • Positions for each match • Average position
Bugs	None known
Implementation Details	WebPosition gathers information about how a site is ranked on search engines. It looks like the only way to get information out of it is through scraping text files that it generates. It only runs on NT. WebPosition is running on web-nt7. URLs to get the data is in the format: web-nt7.xor.com/reports/www.sitename.com.txt
Screenshot	Search Engines **Matches: 91 (34.21%)** As of May 23 2001, 8:43am: ⇳7 d In Top 10: 46 (17.29%) Searches: 266 (19 sites) Avg Position: 12.7 URLs Found: 44

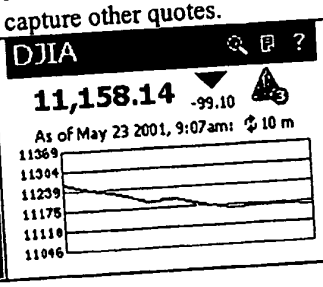
Yahoo News

Language	Perl
Version	1.0
Reusability Level	High, but this probably isn't legal
Customers Using	None

Appendix - 10

Typically Updated	Hourly
Collection Method	HTTP calls, scraping the returned HTML
Statistics Returned	Current news for a variety of topics (World, Tech, Finance, etc)
Bugs	None known
Implementation Details	Calls URLs for different types of news. Examples: Headlines: http://dailynews.yahoo.com/headlines/ts/ Business: http://dailynews.yahoo.com/headlines/bs/ It the scrapes the top news stories out of the HTML.

Yahoo Stocks

Language	Perl
Version	1.1
Reusability Level	High, but this probably isn't legal
Customers Using	TSN
Typically Updated	10 minutes
Collection Method	HTTP calls, scraping the returned HTML
Statistics Returned	Today's history of the following indexes: NYSE, NASDAQ, DOW, and S&P 500. Can easily be customized to get stats for specific stocks
Bugs	The first quote of the day isn't recorded in the reaper repository. .This is likely a bug with the reaper when it receives a "clear history" instruction.
Implementation Details	Call this URL to get stock information: http://quote.yahoo.com/quotes?SYMBOLS=^DJI,^IXIC,^SPX Note that other symbols can be added to the end to capture other quotes.
Screenshot	

Agents Under Development

HP Openview

Language	Perl
Priority	High
Reusability Level	High
Customers	MCC, dashboards for clients
Typically Updated	10 minutes
Description	HPOV monitors machines/equipment from a networking level. This mostly determines if a machine or piece of networking hardware is available or not.

Proposed Agents

Keynote

Likely Language	Perl
Reusability Level	High
Customers	Qwest
Typically Updated	Hourly
Description	Keynote is a site monitoring service, similar to ServiceMetrics.

cPulse

Description	XOR's clients use cPulse to deliver customer satisfaction surveys.

MRTG

Likely Language	Perl
Priority	Probably being scrapped in favor of OpenView stats
Reusability Level	High
Customers	
Typically Updated	10 minutes
Collection Method	Text file scraping
Statistics Returned	Current, average, 95th percentile of bandwidth
Description	MRTG gathers live bandwidth statistics via the hosting infrastructure's switches.

Appendix B

Dashboard Data Types

dataitem

Description	The base class from which nearly all other data types inherit their default behavior.
Notes	Abstract class
Attributes	
name	The name of this particular piece of information. The name can only contain the following characters: A-Za-z0-9_.- (no spaces or punctuation!)
description	A description for this piece of information
type	The data type of this particular piece of information
timestamp	The time the current value was sampled
XML	None. The raw *dataitem* type should not be used directly.

image (isa dataitem)

Description	A pointer to an image URL
Notes	This has yet to be implemented in the controller
Attributes	
image	The URL of an image to display
XML	<image name="status_image" image="http://visionport.xor.com/images/ok.gif"/>

text (isa dataitem)

Description	Holds plain text. The plain text may contain newlines and leading spaces for formatting purposes.
Notes	The formatting may be significant and should be respected by the GUI if possible. The Controller may reformat the text for display on a given device.
Attributes	
text	The text to display
XML	<text name="first_name">Tom</text>

html (isa dataitem)

Description	Holds HTML markup text.
Notes	The Controller may need to reformat the text for display on a given device. The contents must be quoted so as to be valid XML. Alternatively, a CDATA section could be used.
Attributes	
html	The html to display
XML	<html name="fancy_text">this is bold</html>

number *(isa dataitem)*

Description	The base class for all data items that store numbers. It can hold either a single value or a value along with a set of historical values.
Notes	Can hold any number (double/long/fraction/positive/negative). The datasets (if any) embedded into the number will always have names of 'values', 'rates'. and 'timestamps'. The "value" attribute will always store the current value in the array of historical values.
Attributes	
value	The last sampled value of the item
units	What the number represents (dollars, percent, seconds, etc.)
history	Number of values to keep (0 is assumed if absent). This can also be configured in the reaper.
values	*Dataset* of historical values; this is typically automatically created by the reaper
timestamps	*Dataset* of historical values; this is typically automatically created by the reaper
rate	Rate of change based on the *rate_type*; the rate is typically created by the reaper
rates	*Dataset* of historical rates based on the *rate_type*; the rates are typically created by the reaper; this is typically automatically created by the reaper
rate_type	Tells the reaper the type of rate to generate. This is described in detail in the "Reaper" portion of this document, and is typically set in the reaper configuration file.
XML	Minimal: <number name='load_avg' timestamp='970169755' value='1.26'/> Everything: <number name='price' timestamp='970169746' value='1.49' units='$' history='4' rate='-.5' rate_type='average:1'> <dataset name='values'>1.49, 2.01, 1.67, 1.80, 1.57</dataset> <dataset name='rates'>-.5, .34, -.13, .33, .57</dataset> <dataset name='timestamps'>970169750, 970169745, 970169740, 970169735, 970169730</dataset> </number>

counter *(isa number)*

Description	A number that is non-negative and monotonically increasing.
Notes	Only increases until it is reset. If it ever decreases the reaper should throw an exception.
Attributes	
N/A	Same as *number*.
XML	<counter name='page_views' value='389222'

	timestamp='970169755'/>

gauge (isa number)

Description	A non-negative number that indicates some value that can fluctuate up and down (for example, system load or number of active users).
Notes	If the maximum or the minimum is not explicitly specified, then they are unbounded.
Attributes	
maximum	Known maximum value possible; reaper should throw an exception if it increases over this limit.
minimum	Known minimum value possible; reaper should throw an exception if it goes under this limit.
XML	<gauge name='active_users' timestamp='970169755' value='17' history='4'> <dataset name='values'>17, 19, 21, 16, 20</dataset> <dataset name='timestamps'>970169755, 970169745, 970169735, 970169725, 970169715, </dataset> </gauge>

elapsed (isa number)

Description	Elapsed time in seconds, used to express relative time (for example, uptime or visit length).
Notes	Is converted by the Controller into an appropriate format (for example, HH:MM:SS)
Attributes	
N/A	
XML	<elapsed name='uptime' value='38921' timestamp='970169755'/>

chart (isa dataitem)

Description	A "chart" can store a set of axis/data labels plus multiple sets of values. Used instead of a *number* object when the axis is not time-based.
Notes	The number of items in the labels array, and in each values array must match the count attribute.
Attributes	
count	The number of values on the x-axis and the number of values in each *dataset*
labels	Array of labels for the x-axis
dataset(s)	*Dataset(s)* of y-values
XML	<chart name='fuel_prices' count='3'> <labels>oil, gas, coal</labels> <dataset name='price'>35, 50, 15</dataset> <dataset name='tax'>20, 10, 12</dataset> </chart>

labels *(isa dataitem)*

Description	Holds a list of labels (used in *chart*)
Notes	
Attributes	
labels	Array of textual or numeric labels
XML	See *chart*.

dataset *(isa dataitem)*

Description	Stores a list of values (see *chart* and *number* for example use)
Notes	
Attributes	
values	Array of y-values
XML	<dataset name='price'>35, 50, 15</dataset>

list *(isa dataitem)*

Description	Holds a list of other items.
Notes	The name of each item in the list must be unique.
Attributes	
N/A	
XML	<list name='status'> <text name='summary'>All servers are operational</text> <list name='www1'> <number .../> <counter .../> <text name='status'> ... </text> </list> <list name='www2'> ... </list> </list>

instruction *(isa dataitem)*

Description	Communicates an instruction back to the reaper.
Notes	
Attributes	
name	If the 'name' of an instruction data item is "noop", then the reaper will not record any information sent to it. Likewise, it will not update the item's history. This is useful for updating data items that may not change frequently or have a period of inactivity (such as the stock market). If the 'name' of an *instruction* data item is "clear_history", the reaper will clear this data item's history datasets before inserting the current data. This is useful for "resetting" histories. For example, the stock market history could be

[redacted]	cleared every morning when the market opens, so that a new graph is started for the day.
[redacted]	<instruction name='noop'/> <instruction name='clear_history'/>

Appendix C

NAME

Dashboard

DESCRIPTION

Driver class for Dashboard XML project.

SYNOPSIS

Creating (aka. freezing)

```
my $dashboard=new Dashboard();
```

Reading (aka. thawing)

```
my $dashboard=new Dashboard(xml => $xml_string);
my $dashboard=new Dashboard(file => $file_name);
```

Accessing

```
$item=$dashboard->select('path/to/item/within/tree');
```

PREREQUISITES

XML::Parser

CONSTRUCTORS

The constructor can be called two ways: No arguments, which creates a brand-spankin' new data object, or with a chunk of XML or a file name, which will read and parse (``thaw'') the XML, and construct a Dashboard data structure which can be queried, modified, and re-frozen as desired.

```
Dashboard::new(xml => 'some string of XML');
```

or

```
Dashboard::new(file => 'some_filename_containing_XML');
```

METHODS

select("selector/string/path")

This will search the tree for an item that can be reached by the specified list of ``selector" strings, which are separated by ``/" characters.
At each level in the tree, an item will be searched for that matches the given selector. For Lists, the DataItem with the specified name will be returned. For other item types, an attribute of the given name will be searched for. This attribute may be a scalar value, or it might be an Array or a pointer to another DataItem type.
Examples:
Given the following XML:

```
<dashboard>
  <list name="alpha">
    <gauge name='bravo' timestamp='970594169' value='111'/>
    <list name='charlie'>
      <counter name='delta' historyLength='3' timestamp='970594170'
               value='20'>
        <dataset name='values'>20, 30, 40</dataset>
        <dataset name='timestamps'>970594163,970594164,
                 970594165</dataset>
      </counter>
      <elapsed name='echo' value='2932' timestamp='970594168'/>
    </list>
  </list>
  <number name='foxtrot' timestamp='970594179' value='333'/>
</dashboard>
```

The following selectors will return the indicated objects (or die with an error as indicated):

alpha	List named "alpha"
bogus	die: 'bogus' not found
alpha/bravo	Gauge named "bravo"
alpha/bravo/value	Scalar:111
alpha/bravo/value/bogus/stuff	die: leaf at 'bogus'
alpha/bravo/bogus/stuff	die: 'bogus' not found
alpha/bravo/timestamp	Scalar:970594170
alpha/charlie	List named "charlie"
alpha/charlie/delta	Counter named "delta"
alpha/charlie/delta/value	Scalar:20
alpha/charlie/delta/historyLength	Scalar:3
alpha/charlie/delta/values	Dataset named "values"
alpha/charlie/delta/values/values	ARRAY[20,30,40]
alpha/charlie/delta/values/values/0	Scalar:20
alpha/charlie/delta/values/values/3	die: index '3' not found
alpha/charlie/delta/timestamps	Dataset named "timestamps"
alpha/charlie/delta/timestamps/values/1	Scalar:970594164
alpha/charlie/echo/value	Scalar:2932
foxtrot	Number named "foxtrot"
foxtrot/value	Scalar:333

Strange cases:

- To get the text of a text item use the selector ``.../itemname/text"
- To get the body of an html item use ``.../itemname/html"
- To get the URL of an image item use ``.../itemname/image"
- Charts:

- To get the labels for a chart as an array use ``.../chartname/labels/labels".
- To get a dataset for a chart as an array use ``.../chartname/datasets/0/values".
- To get the name of a chart dataset use ``.../chartname/datasets/0/values".
- To find out how many datasets there are use ``.../chartname/datasets/count".

select_scalar("selector/string/path")

Just like `select()`, except that it errors out if the item returned is not a scalar value.

xml_freeze()

Returns an XML string representation of the data structure.

xml_thaw($xml_string)

Returns a new Dashboard object created by parsing the specified XML stored in a string. Preferred: use the 'xml' argument on the constructor.

add(LIST)

Adds a list of DataItems to the current Dashboard object. Will die if an attempt to add an item named identially to an existing item in the Dashboard is made.

```
$dash=new Dashboard();
$count=new Dashboard::Counter(value=>5);
$dash->add($count);
```

or even:

```
$dash->add(new Dashboard::Counter(value=>5));
```

Appendix D

Controller Configuration Files

Navorg

Description	The navorg element defines the initial structure and contains attributes for defining the renderer and the name of the data configuration file to use.
Notes	
Attributes	
name	Informational name of this display configuration.
aliases	The path to the data configuration file. Optional, defaults to *conf/data.xml*.
layout	The name of the renderer that this user will use by default. The existing renderers are *framed* and *htmltable*. Defaults to *framed*.
templatedir	Template-override directory to use for this file. Can be overridden by *templatedir* specified in a *section*. Defaults to the username.
XML	<navorg name="The Sporting News" aliases="tsn_data.xml" layout="framed"> ... </navorg>

Section

Description	The *section* element defines the layout of a single display page. Each display page can have multiple cells (also called constructs or indicators) on it. *Sections* are linked together via the *onSelect* attributes of *constructs* (see below).
Notes	
Attributes	
name	The name of the section. The section *name* must be unique across all sections. Required. "start" is the *name* of the first page the user will view.
template	Defines the template directory to use for overriding the default template directory. Defaults to the *username*.
templatedir	Template-override directory to use for this file. Defaults to the *templatedir* specified in the *navorg*.
rows	The number of cells per column to create.
cols	The number of cells per rows to create.
rowsize	The height of each row in the display. Defaults to 150.
colsize	The width of each column in the display. Defaults to 200.
description	The description of the section. This is not available to the templates and is mainly for documentation purposes.

	information.
onSelect	Defines the *section name* that the indicator's drilldown icon (magnifying glass) will be associated with. If not specified, the indicator will not have a drilldown icon.
onSelect ScreenType	Defines if the drilldown will be displayed in a separate popup window ("popup"), replaces the full screen ("full"), or replaces the current frame ("frame"). Defaults to "popup".
precision	Specifies a precision to limit all number elements to. The default is to leave numbers as-is. A precision of "1" will make numbers look like "2.5". This is handy for limiting the precision of averaged aggregate items.
XML	<construct name="Control" description="Control" type="control" position="1"> </construct> <construct name="DJIAcurrent" description="DJIA" type="number" position="2" onselect="DJIA_graph" data_refresh="10"> <attr name="helptext"> <U><B>Dow Jones Industrial Average</B></U> <P> The large blue number is the current value of the DJIA. The arrow indicates the current direction of the market since the market last opened. The arrow will be green if the market is up for the day, and red if it's down. The small number under the arrow is the change since the market last opened. <P> The chart displays how the market has moved over the past 8 hours of activate trading. Every morning when trading begins, the chart begins anew. <P> This indicator is updated every 10 minutes. </attr> </construct>

Item

Description	Item elements define the set of data to be made available for use in the display template. An *item* can only appear inside a set of *construct* tags.
Notes	
Attributes	
name	The name of the item. The *name* will in most cases also defines the name of the variable available in the template. An item *name* must be unique across the enclosing *construct*.
source	The alias name of the source object (from *data.xml*).
select	An additional path into the *source* item's object tree.

	Depending on the template variable that's being populated, the select will be either a leaf node (such as /sitestats/hits/value), or a non-leaf (such as /sitestats/hits/history). If a non-leaf is required for the template, the controller will do the work of gathering the required information out of the node.
format	Sets a format for the controller to use for this item. Valid *formats* are: • dollars (uses precision of 2 and pre-pends a "$") • elapsed (converts to *DD* days, *HH:MM*) The controller also automatically formats: • If the *select* contains "timestamp", it will change the format to *MM/DD/YY HH:MM*. This is primarily used for auto-populating the x-axis on graphs. • If the item looks like a number (contains only digits and periods), it will: make a default precision of 2 and "comma-ize" the number. For example, 2003.456 will be changed to 2,003.46.
precision	Sets the numeric precision for the item.
type	Defines a type of the particular item, which affects what is done with the data. Valid *types* are: • graph • table • var • alert Each of these *types* is explained in detail below.
unit	Sets the unit string for the data item. The *unit* value is made available to the template in the *unit* variable.
prefix	Defines a prefix that is typically displayed before the item. The *prefix* value is made available to the template in the *prefix* variable.
graphtype	The type of graph to display (only valid with a *type* of "graph"). Can be one of: • bars • pie • lines Defaults to "lines" if not specified.
value	Sets the value of an *item* of *type* "var".
XML	<construct name="SysLoad" description="System Load" type="number" position="14"> <item name="alert" type="alert" source="ALERTS" select="SysLoad"/> <item name="graph" type="graph" source="systemload" select="load15"/> <item name="current" source="systemload" select="load15/value"/>

	<item name="history" source="systemload" select="load15/values/values"/> <item name="direction" source="systemload" select="load15/rate"/> <item name="timestamp" source="systemload" select="load15/timestamp"/> </construct>

Item Types

var

Description	Defines a variable to set in the template.
Notes	
Attributes	
name	The name of the variable.
value	The value to set this variable to.
XML	<construct name="DJIA" description="DJIA Table" type="number" template="labels_only" position="1"> <item name="label1" type="var" value="Time"/> <item name="label2" type="var" value="Value"/> <item name="label3" type="var" value="Change"/> <item name="label4" type="var" value="% Change"/> </construct>

alert

Description	If an alert it set for the *source/select* defined, then the *name* is used to populate a set of variables. If the *name* is set to "alert", then the following variables will be defined (based off of how the alert was set up in the alerts configuration file): • alert_severity: The severity of the alert (0 through 3) • alert_msg: The alert message to display to the user • alert_notes: Notes (not currently used) • alert_level: The name of the alert level • alert_directional: Directional graphic suffix (e.g.: "red")
Notes	
Attributes	
name	The prefix of a template variable to populate.
source	The source file for the alerts.
select	The data item in the alerts file to look for.
XML	<construct name="DJIA" description="DJIA" type="number" position="1"> <item name="alert" type="alert" source="ALERTS" select="DJIA"/> </construct>

table

Description	Creates a table based on data elements supplied to it.
Notes	The current implementation can only grab data from one

	<item name="history" source="systemload" select="load15/values/values"/> <item name="direction" source="systemload" select="load15/rate"/> <item name="timestamp" source="systemload" select="load15/timestamp"/> </construct>

Item Types

var

Description	Defines a variable to set in the template.
Notes	
Attributes	
name	The name of the variable.
value	The value to set this variable to.
XML	<construct name="DJIA" description="DJIA Table" type="number" template="labels_only" position="1"> <item name="label1" type="var" value="Time"/> <item name="label2" type="var" value="Value"/> <item name="label3" type="var" value="Change"/> <item name="label4" type="var" value="% Change"/> </construct>

alert

Description	If an alert it set for the *source/select* defined, then the *name* is used to populate a set of variables. If the *name* is set to "alert", then the following variables will be defined (based off of how the alert was set up in the alerts configuration file): • alert_severity: The severity of the alert (0 through 3) • alert_msg: The alert message to display to the user • alert_notes: Notes (not currently used) • alert_level: The name of the alert level • alert_directional: Directional graphic suffix (e.g.: "red")
Notes	
Attributes	
name	The prefix of a template variable to populate.
source	The source file for the alerts.
select	The data item in the alerts file to look for.
XML	<construct name="DJIA" description="DJIA" type="number" position="1"> <item name="alert" type="alert" source="ALERTS" select="DJIA"/> </construct>

table

Description	Creates a table based on data elements supplied to it.
Notes	The current implementation can only grab data from one

	source. Labels should also be moved down into options, like "legend" is for the "graph" type.
Attributes	
name	The name of the table. The *name* is the name of the variable to be used in the template's loop.
source	The source file for the table data.
option	*Option*s are specified as elements of a *table*. Each *option* defines one data source for the table. The *option* contains a *name* attribute that defines the variable to populate in the template's loop). The body of the *option* contains the select string into the *table*'s data *source*.
XML	<construct name="DJIA" description="DJIA Table" type="number" template="detail_table" position="1"> <item name="label1" type="var" value="Time"/> <item name="label2" type="var" value="Value"/> <item name="label3" type="var" value="Change"/> <item name="label4" type="var" value="% Change"/> <item name="table" type="table" source="DJIA"> <option name="data1">current/timestamps/values</option> <option name="data2">current/values/values</option> <option name="data3">difference/values/values</option> <option name="data4">percent/values/values</option> </item> </construct>

graph

Description	Creates a graph based on data elements supplied to it.
Notes	The current implementation can only grab data from one *source*.
Attributes	
name	The name of the graph. The *name* is the name of the variable to be used in the template's loop.
source	The source file for the graph data.
select	An additional path into the *source* item's object tree.
option	*Option*s specify a number of arguments to the graph: • *axis*_data*num*: If an option name has the format: x_data*num* or y_data*num* (where *num* is a data plot number, starting with "1"), then the body of the *option* contains the select string into the *table*'s data *source*. If these types of *option*s are not specified, then the graph's *source* is used as a single element to graph. The x-axis is the *source*'s timestamps, and the y-axis is the *source*'s values. • label*num:* A label for the specified *num* data set. The default is to not have any labels on the graph. • width: The width of the graph. The default is 180. • height: The height of the graph. The default is 120.

	• bgclr: The background color of the graph. The default is yellow (255,255,204).
XML	<construct name="HPLoad" description="HP Load Time" type="number" position="13"> <item name="graph" type="graph" source="servicemetrics"> <option name="width">500</option> <option name="height">200</option> <option name="bgclr">#ffffff</option> <option name="x_data1">site1/values/values</option> <option name="y_data1">site1/timestamps/values</option> <option name="legend1">Site 1</option> <option name="x_data2">site2/values/values</option> <option name="y_data2">site2/timestamps/values</option> <option name="legend2">Site 2</option> </item> </construct>

Appendix E

Dashboard Controller Variables

Cause	An element's name is equal to "timestamp" or an element's *select* contains "timestamp".
Effect	Populates the following variables, with the element's *name* as the prefix: • *prefix*_sec: Seconds • *prefix*_min: Minutes • *prefix*_hour: Hour in 24-hour clock • *prefix*_mday: Day of the month • *prefix*_mon: Month number • *prefix*_year: 4-digit year • *prefix*_mon_name: 3-character month name • *prefix*_time: 12-hour time (e.g.: 1:23 PM) • *prefix*_date: Date in MM/DD/YY format (e.g.: 1/13/01)
Notes	A timestamp typically indicates a refresh time of the data object. It is supplied in the Unix time format (seconds).

Cause	The "description" attribute is populated (which is required for all elements anyway).
Effect	Populates a "description" variable.

Cause	A "graph" item exists in the construct.
Effect	A "graph" variable is populated with the URL to generate the graph image. A "graph_map" variable is populated with the graph's imagemap. A "width" variable is populated with the graph's width. A "height" variable is populated with the graph's height.

Cause	The "refresh" or "data_refresh" attributes are populated.
Effect	Populates a "refresh" variable that contains the refresh rate in the following format: • 10 s (10 seconds) • 10 m (10 minutes) • 10 h (10 hours) • 10 d (10 days) The controller figures out which format to use depending on how large the refresh number is.
Notes	This currently doesn't deal with fractions (i.e.: 1.6 h is seen as 1h).

Cause	None
Effect	A "script" variable is populated with the URL that generated the current construct. This doesn't have much use in the templates, but it's used internally in the controller.

Cause	None
Effect	A "dashboardtop" variable is populated with the URL that generates the top-level of the dashboard. Useful for linking a "home" button.

Cause	onSelectScreen attribute is set in the *construct*.
Effect	One of "onSelectScreenFull", "onSelectScreenFramed", or "onSelectScreenPopup" variables is set to 1, depending on the value of *onSelectScreen*.

Cause	onSelect attribute is set in the *construct*.
Effect	"drilldown", "drilldownframed", "drilldowntable" variables are populated with the URL to the selected section (current format, framed format and table format, respectively). The URLs are auto-generated based on the onSelect attribute in this item's *construct*.

Cause	BUG
Effect	An "onReport" attribute should be recognized inside the *construct*, but currently it's not. Instead, we set it using a "var" type, which works out just fine.

Cause	unit attribute is populated within an *item*.
Effect	An *itemname*_unit variable is populated with its value.
Notes	This could probably be moved into an *item* of *type* "var".

Cause	prefix attribute is populated within an *item*.
Effect	An *itemname*_prefix variable is populated with its value.
Notes	This could probably be moved into an *item* of *type* "var".

Cause	An *item* named "direction" exists.
Effect	If the value of "direction" is: • positive: Variable "directionup" is set to 1 • zero: Variable "directionneutral" is set to 1 • negative: Variable "directiondown" is set to 1
Notes	This is useful for determining if an up arrow or a down arrow should be displayed on the indicator. A direction item is typically the indicator's rate: <item name="direction" source="sysload" select="cpu/rate"/>

Cause	An *item* named "history" exists.
Effect	The following variables are set: • "high": Highest value in the history • "low": Lowest value in the history

Cause	An attribute named "helptext" exists.
Effect	A "helptext" variable is populated and a "helpurl" variable is populated with the URL to a help popup window

Cause	An "alert" type exists within the *construct*.
Effect	See the "alert" type above.

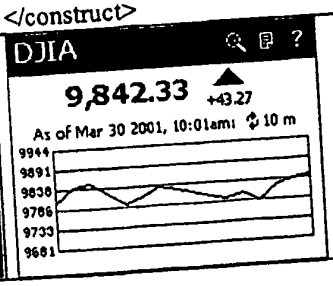
Appendix F

Dashboard Controller Templates

Number

Description	Displays a small, framed primary indicator with a graph beneath it of the indicator's history
Notes	This should really be renamed to "summary" instead of "number".
Primary Template	htmltable/number.tmpl
Templates Included	htmltable/visionport.css htmltable/visionport.js htmltable/summary_titlebar.tmpl htmltable/summary_numbdisp.tmpl htmltable/summary_asofdate.tmpl htmltable/summary_graph.tmpl
Variables Used	
Titlebar	
Description	Used as the title for the indicator
drilldown table	If populated, a detail icon will be created with a link to the URL contained in *drilldowntable*. Note that this variable is automatically created by the controller if *onSelect* is populated.
drilldown framed	If populated, and *onselectscreenframe* is also populated, a detail icon will be created with a link to the URL contained in *drilldownframed*. Note that this variable is automatically created by the controller if *onSelect* is populated.
onSelect ScreenFull	If populated, and *drilldownframed* is populated, a detail icon will be created with a link to the URL contained in *drilldownframed* with target="_top". Note that this variable is automatically created by the controller if *onSelectScreen* is populated.
onSelect ScreenFrame	If populated, and *drilldownframed* is populated, a detail icon will be created with a link to the URL contained in *drilldownframed*, to appear in the existing frame. Note that this variable is automatically created by the controller if *onSelectScreen* is populated.
reporturl	If populated, a report icon link will be created to the URL contained in *reporturl*.
onReport ScreenFull	If *reporturl* is populated, adds target="_top" to the link. By default, a *reporturl* is displayed in a popup window.
onReport ScreenFrame	If *reporturl* is populated the *reporturl* will be displayed in the current frame. By default, a *reporturl* is displayed in a popup window.
helpurl	If populated, a help icon link will be created to the URL

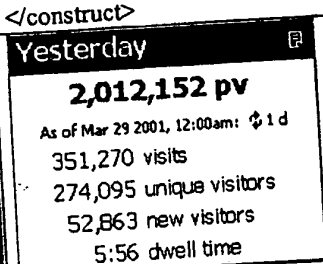
	contained in *helpurl*. Note that this variable is created automatically by the controller if *helptext* is populated.
Numbdisp	
current	The primary indicator number, displayed in bold, as populated by an *item* named "current".
current_unit	If a *unit* is specified within the "current" *item*, it will be placed directly after the primary indicator (e.g.: 123 page views).
direction_up	One of *directionup*, *directiondown*, or *directionneutral* is set by the controller depending on the value specified by the *difference* item.
direction_down	See *directionup*.
direction_neutral	See *directionup*.
alert_directional	*alert_directional* is set if an alert is found for this indicator. The directional is used as a suffix for the up/down image. For example, if *alert_directional* is set to "red", then "upred.gif" or "downred.gif" will be leaded as the up/down image.
alert_severity	*alert_severity* is set if an alert is found for this indicator. The severity is used as a suffix for the alert icon. For example, if *alert_severity* is set to "2", then "alert1.gif" is used for the alert image. If *alert_severity* is 0, then no alert image is displayed. *alert_severity* is also displayed when the user mouses over the alert icon.
alert_msg	*alert_msg* is set if an alert is found for this indicator. The msg is displayed along with the alert's severity when the user mouses over the alert icon.
direction	The current "direction" of the indictor, as populated by an *item* named "direction". This is typically the current rate of change.
direction_unit	If a *unit* is specified within the "direction" *item*, it will be placed directly after the direction (e.g.: -3.4%).
Asofdate	
timestamp_mon_name	If an *item* named "timestamp" is populated, the controller will automatically populate these timestamp_* variables. These variables are used to tell the end-user when the last time the data was updated.
timestamp_mday	See *timestamp_mon_name*.
timestamp_year	See *timestamp_mon_name*.
timestamp_time	See *timestamp_mon_name*.
data_refresh	If the *construct* contains a *data_refresh* attribute, or the

	construct contains a *screen_refresh item*, then this variable will automatically be populated by the controller. This variable is used to tell the user how frequently the data is updated.
Graph	
graph	If a "graph" *item* is contained in the *construct*, then the controller will automatically populate this variable. The variable contains a link to the URL that will display the graph.
drilldown table	If this variable is populated (see the "Titlebar" section), then the graph is also linked to this URL.
description	The *description* is used to populate the ALT tag of the graph.
XML	<construct name="DJIAcurrent" description="DJIA" type="number" position="2" onselect="DJIA_graph"> <item name="reporturl" type="var" value="http://quote.yahoo.com/q?s=^dji&d=b"/> <item name="alert" type="alert" source="ALERTS" select="DJIA"/> <item name="alert_per" type="alert" source="ALERTS" select="DJIA_percent"/> <item name="graph" type="graph" source="DJIA" select="current"/> <item name="current" source="DJIA" select="current/value"/> <item name="history" source="DJIA" select="current/values/values"/> <item name="direction" source="DJIA" select="difference/value"/> <item name="percent" source="DJIA" select="percent/value"/> <item name="timestamp" source="DJIA" select="current/timestamp"/> <item name="screen_refresh" source="DJIA" select="current/refresh"/> <attr name="helptext"> ... </attr> </construct>
Screenshot	

Appendix - 33

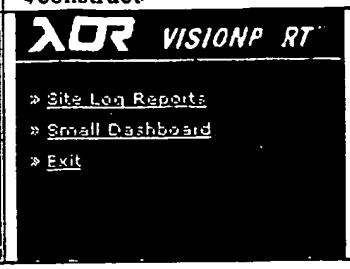
Summary_stats

Description	Displays a small, framed primary indicator with a set of supporting statistics beneath it.
Primary Template	htmltable/summary_stats.tmpl
Templates Included	htmltable/visionport.css htmltable/visionport.js htmltable/summary_titlebar.tmpl htmltable/summary_statupper.tmpl htmltable/summary_asofdate.tmpl htmltable/summary_statslower.tmpl
Variables Used	
Titlebar	See the "Titlebar" section under the Number template.
statsupper	
val1	The primary indicator number, displayed in bold, as populated by an *item* named "val1".
val1_unit	If a *unit* is specified within the "val1" *item*, it will be placed directly after the primary indicator (e.g.: 123 page views).
val1_prefix	If a *prefix* is specified within the "val1" *item*, it will be placed directly before the primary indicator (e.g.: Page Views: 123).
val1rate	The rate of change for the primary indicator, as populated by an *item* named "val1rate".
val1rate_unit	If a *unit* is specified within the "val1rate" *item*, it will be placed directly after the rate (e.g.: -1.23%).
Asofdate	See the "Asofdate" section under the Number template.
statslower	
valx	A statistic, as populated by an *item* named "valx", where *x* is a number from 2 to 5.
valx_unit	If a *unit* is specified within the "valx" *item*, it will be placed directly after the primary indicator (e.g.: 123 page views).
valx_prefix	If a *prefix* is specified within the "valx" *item*, it will be placed directly before the primary indicator (e.g.: Page Views: 123).
valxrate	The rate of change for the primary indicator, as populated by an *item* named "valxrate".
valxrate_unit	If a *unit* is specified within the "valxrate" *item*, it will be placed directly after the rate (e.g.: -1.23%).
XML	<construct name="SiteStats_yesterday" description="Yesterday" type="summary_stats" data_refresh="86400" position="9"> <item name="reporturl" type="var" value="/cgi-bin/webtrends_redirect/5/1"/> <item name="val1" source="historical_stats" select="all/yesterday/pageviews/value" unit="pv"/> <item name="timestamp" source="historical_stats" select="all/yesterday/pageviews/timestamp"/> <item name="screen_refresh" source="historical_stats"

	select="all/yesterday/pageviews/refresh"/> <item name="val2" source="historical_stats" select="all/yesterday/visits/value" unit="visits"/> <item name="val3" source="historical_stats" select="all/yesterday/unique_visitors/value" unit="unique visitors"/> <item name="val4" source="historical_stats" select="all/yesterday/firsttime_visits/value" unit="new visitors"/> <item name="val5" source="historical_stats" select="all/yesterday/dwell/value" unit="dwell time" format="elapsed"/> </construct>
Screenshot	

Control, Control_subpage

Description	Displays a small, framed "Control Panel". The only difference between Control and Control_subpage is the inclusion of a "back" button on the subpage version.
Primary Template	htmltable/control.tmpl htmltable/control_subpage.tmpl
Templates Included	htmltable/visionport.css htmltable/visionport.js
Variables Used	
dashboard_top	Link to the "top level" of the dashboard. This variable is automatically populated by the controller.
weblog	If populated, the *weblog* variable will link the user to a 3rd party product for viewing their web logs.
weblog_desc	Description shown to the user if *weblog* is populated.
weblog2	Second *weblog*, typically populated if the user has two weblog profiles to view.
weblog_desc2	Description for the second *weblog*.
XML	<construct name="Control" description="Control" type="control" position="1"> <item name="weblog" type="var" value="http://visionport.xor.com/webtrends/demo/"/> <item name="weblog_desc" type="var" value="Site Log Reports"/>

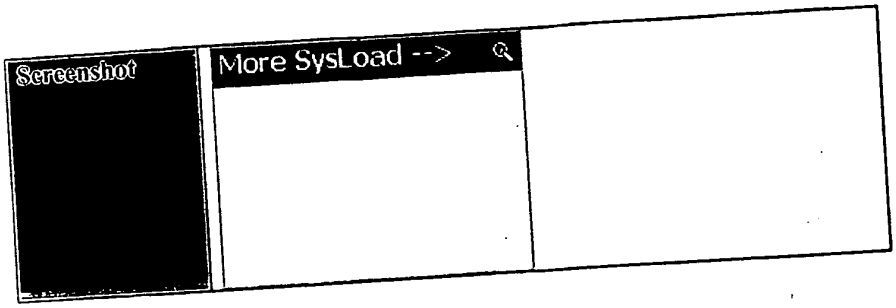
	</construct>
Screenshot	

Clientlogo

Description	Displays a small, framed logo.
Primary Template	htmltable/clientlogo.tmpl
Variables Used	
image	URL to the image to display.
width	Width of the image.
height	Height of the image.
alt	ALT text for the image.
url	Link to the customer's web site
XML	<construct name="Logo" description="Logo" type="clientlogo" position="16"> <item name="image" type="var" value="http://www.client.com/images/logo.gif"/> <item name="width" type="var" value="100"/> <item name="height" type="var" value="50"/> <item name="alt" type="var" value="Client Site"/> <item name="url" type="var" value="http://www.client.com"/> </construct>

Blank

Description	Displays only a title bar. This is used for displaying an indicator that only has links to a detail and/or report icon. Useful for displaying a "More Indicators" cell.
Primary Template	htmltable/blank.tmpl
Templates Included	htmltable/visionport.css htmltable/visionport.js htmltable/summary_titlebar.tmpl
Variables Used	
Titlebar	See the "Titlebar" section under the Number template.
XML	<construct name="next" description="More SysLoad -->" type="blank" position="2" onselectscreen="full" onselect="SysLoad_graph2"> </construct>

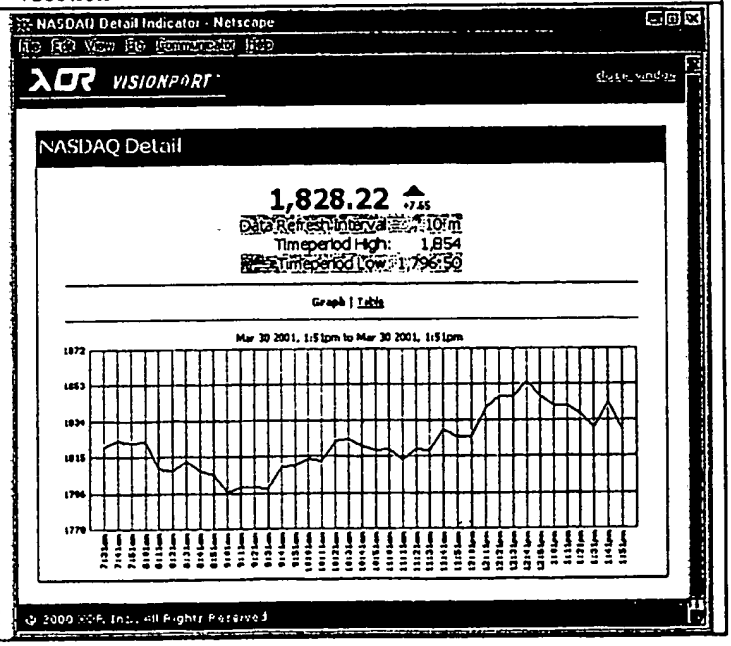
Screenshot	

Detail_graph

Description	Displays a larger version of a "Number" template. The graph is significantly larger and contains labels on the x- and y-axis. High and lows from the graph are also displayed.
Primary Template	htmltable/detail_graph.tmpl
Templates Included	htmltable/visionport.css htmltable/visionport.js htmltable/detail_titlebar.tmpl htmltable/detail_numbdisp.tmpl htmltable/detail_graph_toggle.tmpl htmltable/detail_graph_body.tmpl htmltable/detail_footer.tmpl
Titlebar	
description	Used for the title of the page.
Numbdisp	See the "Numbdisp" section under the Number template. The following variables are also used:
data_refresh	If the *construct* contains a *data_refresh* attribute, or the *construct* contains a *screen_refresh item*, then this variable will automatically be populated by the controller. This variable is used to tell the user how frequently the data is updated.
high	The *high* is automatically populated if the *construct* has a "history" *item* defined.
low	The *low* is automatically populated if the *construct* has a "history" *item* defined.
Graph_toggle	
Drilldown table	Used to toggle between a graph and a table. The *onSelect* should be set to the table version of this graph.
Graph_body	See the "graph" section of the Number template. Also uses the timestamp variables *starttime* and *enddtime*.
Footer	No variables; just an HTML footer with XOR's copyright.
XML	<section name="NASDAQ_graph" description="NASDAQ Detail" template="tsn"> <construct name="NASDAQ" description="NASDAQ Detail" type="number" position="1" onselect="NASDAQ_chart" template="detail_graph"> <item name="graph" type="graph" source="NASDAQ"

```
select="current">
    <option name="width">500</option>
    <option name="height">200</option>
    <option name="bgclr">#ffffff</option>
  </item>
  <item name="current" source="NASDAQ"
select="current/value"/>
  <item name="history" source="NASDAQ"
select="current/values/values"/>
  <item name="direction" source="NASDAQ"
select="difference/value"/>
  <item name="percent" source="NASDAQ"
select="percent/value"/>
  <item name="starttime" source="NASDAQ"
select="current/timestamps/values"/>
  <item name="endtime" source="NASDAQ"
select="current/timestamp"/>
  <item name="timestamp" source="NASDAQ"
select="current/timestamp"/>
  <item name="screen_refresh" source="NASDAQ"
select="current/refresh"/>
 </construct>
</section>
```

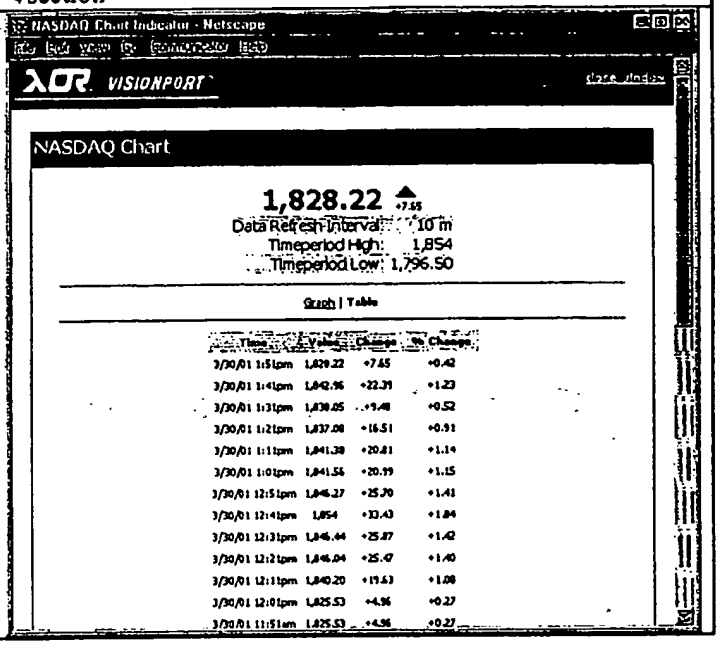
Screenshot



Detail_table

Description	Displays a table of the raw data for a set of information. This is typically linked with a detail_graph template.

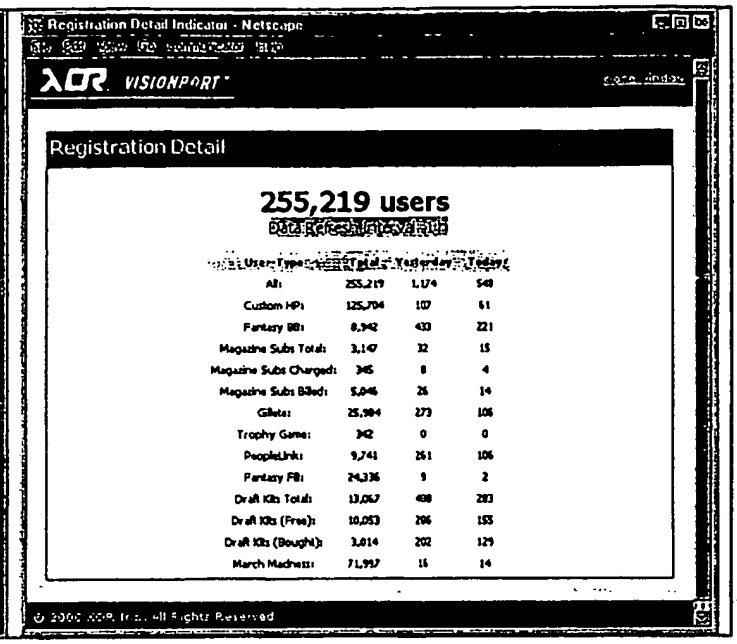
Primary Template	htmltable/detail_table.tmpl
Templates Included	htmltable/visionport.css htmltable/visionport.js htmltable/detail_titlebar.tmpl htmltable/detail_numbdisp.tmpl htmltable/detail_table_toggle.tmpl htmltable/detail_table_body.tmpl htmltable/detail_footer.tmpl
Variables Used	
Titlebar	See the "Titlebar" section under the Detail_graph template.
Numbdisp	See the "Numbdisp" section under the Detail_graph template.
Table_toggle	See the "Graph_toggle" section under the Detail_graph template.
Table_body	
labelx	Labels for the chart, where *x* is 1 through 5
datax	Data items that are looped over to populate the chart. *x* is a number 1 through 5.
Footer	No variables; just an HTML footer with XOR's copyright.
XML	<section name="NASDAQ_chart" description="NASDAQ Chart" template="tsn"> <construct name="NASDAQ" description="NASDAQ Chart" type="number" template="detail_table" position="1" onselect="NASDAQ_graph"> <item name="label1" type="var" value="Time"/> <item name="label2" type="var" value="Value"/> <item name="label3" type="var" value="Change"/> <item name="label4" type="var" value="% Change"/> <item name="table" type="table" source="NASDAQ"> <option name="data1">current/timestamps/values</option> <option name="data2">current/values/values</option> <option name="data3">difference/values/values</option> <option name="data4">percent/values/values</option> </item> <item name="current" source="NASDAQ" select="current/value"/> <item name="history" source="NASDAQ" select="current/values/values"/> <item name="direction" source="NASDAQ" select="difference/value"/> <item name="percent" source="NASDAQ" select="percent/value"/> <item name="starttime" source="NASDAQ" select="current/timestamps/values"/> <item name="endtime" source="NASDAQ" select="current/timestamp"/>

	<item name="timestamp" source="NASDAQ" select="current/timestamp"/> <item name="screen_refresh" source="NASDAQ" select="current/refresh"/> </construct> </section>
Screenshot	

Detail_stats

Description	Displays a larger version of a "Number" template. The graph is significantly larger and contains labels on the x- and y-axis. High and lows from the graph are also displayed.
Primary Template	htmltable/detail_stats.tmpl
Templates Included	htmltable/visionport.css htmltable/visionport.js htmltable/detail_titlebar.tmpl htmltable/detail_numbdisp.tmpl htmltable/detail_stats_body.tmpl htmltable/detail_footer.tmpl
Variables Used	
Titlebar	See the "Titlebar" section under the Detail_graph template.

Numbdisp	See the "Numbdisp" section under the Detail_graph template.
Stats_body	
labelx	Labels for the chart, where *x* is 1 through 5
Data_x_y	Data items that are used to populate the table of statistics. *x* and *y* range from 1 to 5. There's probably a much better way to do this than the way it's currently coded…
Footer	No variables; just an HTML footer with XOR's copyright.
XML	<construct name="RegUsersDetail" description="Registration Detail" type="number" template="detail_stats" position="1" data_refresh="3600"> <item name="label1" type="var" value="User-Type"/> <item name="label2" type="var" value="Total"/> <item name="label3" type="var" value="Yesterday"/> <item name="label4" type="var" value="Today"/> <item name="data1_1" type="var" value="All:"/> <item name="data1_2" source="reg_history" select="appusers/all/All/value"/> <item name="data1_3" source="reg_history" select="appusers/yesterday/All/value"/> <item name="data1_4" source="reg_today" select="appusers/today/All/value"/> <item name="data2_1" type="var" value="Custom HP:"/> <item name="data2_2" source="reg_history" select="appusers/all/16/value"/> <item name="data2_3" source="reg_history" select="appusers/yesterday/16/value"/> <item name="data2_4" source="reg_today" select="appusers/today/16/value"/> <item name="data3_1" type="var" value="Fantasy BB:"/> <item name="data3_2" source="reg_history" select="appusers/all/23/value"/> <item name="data3_3" source="reg_history" select="appusers/yesterday/23/value"/> <item name="data3_4" source="reg_today" select="appusers/today/23/value"/> … </construct>

Screenshot	

Show_help

Description	Template used for the popup help window.
Primary Template	htmltable/blank.tmpl
Templates Included	htmltable/visionport.css htmltable/detail_titlebar.tmpl htmltable/detail_footer.tmpl
Variables Used	
helptext	Help text, as defined a *construct*'s *helptext* attribute.
Titlebar	See the "Titlebar" section under the Detail Graphtemplate.
Footer	See the "Footer" section under the Detail Graph template.
XML	N/A
Screenshot	N/A

Small Dashboard

Description	A collection of the summary_* templates that are used to display a smaller version of the dashboard.
Primary Template	
Templates Included	htmltable/small/*.tmpl
Variables Used	See the normal templates for the variables. There is no additional work required in the controller configuration file to use the small templates. The dashboard has a "switch", controlled by the *subtemplate* QUERY_STRING variable on

	the URL. When the controller sees this set, it first looks in the named *subtemplate* directory for the template first, and then defaults to the normal template directory.
XML	N/A
Screenshot	N/A

Appendix G

Dashboard Indicator Details

Indicator	Data Details
DJIA, NASDAQ, S&P 500	The Dow Jones Industrial Average, NASDAQ index and Standard and Poors 500 index. This information is gathered from Yahoo Financial by "screen scraping" the data from their HTML pages. When XOR has a client that wants this information a stock subscription service will be used.
Today's Stats, Today's Non-web	This information is gathered every 10 minutes from the client's different web servers. A process runs on each server and constantly collects information about visits, hits, page views and content groups being served. Note that this currently only runs on Unix systems. This indicator also displays the number of new registrations received for the current day and the number of visits from users that have registered previously on the site. Selecting the "detail" icon displays a breakdown of statistics for each server.
Today's Ads Served	The client that this data is taken from displays banner ads on their site using a 3rd party ad-serving program called RealMedia OpenAdstream. XOR developed a Dashboard agent that parses the log files of the ad-server and reports on each ad "position" being served. Selecting the "detail" icon displays a breakdown of all the different ad positions.
Today's Revenue	The revenue number is a combination of dollars per visitor (this is how the client estimates their advertisement and sponsorship revenue) and actual subscription services sold on the site. Subscription information is extracted from the client's Informix database.
Yesterday, 8 Days ago, 7 Day Average, 28 Day Average, Visits Per Day	These statistics are gathered either from XOR's proprietary log analysis tool (all clients hosted by XOR receive a monthly report generated by this tool) or from another 3rd party log analysis package, such as CommerceTrends. In this case, CommerceTrends was used. Selecting the "report" icon brings the user into the CommerceTrends report for the selected date.

HP Load Time	XOR has a partnership with a Service Metrics, a company that measures download times for a given URL from multiple points around the Internet. This gives the client a good idea about an average user's "download time experience". The graph displays two different views of this client's site: downloading their homepage with ads and without ads. This was done to help pinpoint problems with a 3rd party ad service (24x7 Media) that caused the client's homepage to consistently load at least two seconds slower when using 24x7 Media's service. The third line on the graph is an "Internet Index" that Service Metrics provides. This is the average download time of 100 large Internet sites. There's another index, the "eCommerce Index" that can also be displayed with this graph. Service Metrics is a standard option of the VisionPort offering to which a client can subscribe.
Average System Load	This measures the average system load across the client's four main servers. Typically, the higher the load, the busier the machine. This information is gathered on-demand from each machine's operating system statistics. Selecting the "detail" icon displays system details about each individual machine.
Registered Users	This particular client has a registration system on their site. The indicator displays the number of users that have registered for various services. The main Dashboard screen displays the services that the client is most interested in watching. This indicator is changed at the client's request to display other registration types. Selecting the "detail" icon displays a breakdown of all types of registered users.

What is claimed is:

1. A system for monitoring business performance indicators in a networked environment, comprising:

a data source having a predefined format;

an agent communicatively coupled to the data source, wherein the agent is configured according to the data source format and wherein the agent is operative to gather data from the data source and translate the data into a first modified format thereby creating modified data;

a reaper communicatively coupled to the agent and configured to retrieve the modified data from the agent;

a data repository communicatively coupled to the reaper and configured to store the modified data;

an alert detector communicatively coupled to the data repository and configured to compare the modified data with a first configuration parameter; and

a dashboard controller communicatively coupled to the data repository and configured to display the modified data in a format defined by a second configuration parameter.

2. The system of claim 1, further comprising an agent polling configuration file communicatively coupled to the reaper and configured to store a data polling schedule and provide the data polling schedule to the reaper.

3. The system of claim 1, further comprising an alert configuration file communicatively coupled to the alert detector and adapted to store the first configuration parameter.

4. The system of claim 1, further comprising a visual configuration file communicatively coupled to the dashboard controller and adapted to store the second configuration parameter.

5. The system of claim 1, wherein the dashboard controller comprises an interface for translating the modified data into a user-readable format.

6. The system of claim 5, wherein the dashboard controller further comprises a memory cache.

7. The system of claim 1, wherein the dashboard controller comprises a plurality of interfaces for translating the modified data into a plurality of user-readable formats.

8. The system of claim 1, further comprising a display device communicatively coupled to the dashboard controller and adapted to present the modified data in a user-readable format.

9. The system of claim 8, wherein the user-readable format is Hyper-Text Markup Language.

10. The system of claim 8, wherein the user-readable format is Wireless Markup Language.

11. The system of claim 8, wherein the display device is a monitor.

12. The system of claim 8, wherein the display device is a cellular phone.

13. The system of claim 8, wherein the display device is a pager.

14. The system of claim 1, further comprising a VoiceXML interface communicatively coupled with the dashboard controller.

15. The system of claim 1, wherein the data source is a proprietary data source.

16. The system of claim 1, wherein the data source is a legacy data source.

17. The system of claim 1, wherein the data source is a third-party application.

18. The system of claim 1, wherein the data source resides on a local area network.

19. The system of claim 1, wherein the data source resides on a wide area network.

20. The system of claim 1, wherein the data source is accessible through the Internet.

21. The system of claim 1, wherein the reaper is in two way communication with the agent.

22. The system of claim 1, wherein the alert detector is adapted to send a notification based on the comparison between the modified data and the first configuration parameter.

23. The system of claim 22, wherein the notification is sent via an email message.

24. The system of claim 22, wherein the notification is sent via a pager message.

25. The system of claim 22, wherein the notification is sent via an SNMP trap.

26. The system of claim 22, wherein the notification is sent via an internet browser alert.

27. The system of claim 1, wherein the networked environment is an electronic commerce system.

28. A system for monitoring business performance indicators in a networked environment, comprising:

a data source having a predefined format;

an agent communicatively coupled to the data source, wherein the agent is configured according to the data source format and wherein the agent is

operative to gather data from the data source and translate the data into a first modified format thereby creating modified data;

a reaper communicatively coupled to the agent and configured to retrieve the modified data from the agent;

a repository manager communicatively coupled to the reaper;

a data repository communicatively coupled to the repository manger;

an alert detector communicatively coupled to the repository manager; and

a dashboard controller communicatively coupled to the repository manager.

29. The system of claim 28, wherein the repository manager includes a cache, and wherein the repository manger is configured to manage the storage of the modified data within the data repository.

30. The system of claim 28, further comprising an alert detector communicatively coupled to the repository manager and configured to compare the modified data with a first configuration parameter.

31. The system of claim 28, wherein the repository manager is in two way communication with the reaper.

32. The system of claim 28, wherein the data repository is in two way communication with the repository manager.

33. The system of claim 28, wherein the alert detector is in two way communication with the repository manager.

34. The system of claim 28, wherein the dashboard controller is in two way communication with the repository manager.

35. A system for monitoring a plurality of business metrics in a networked environment, comprising:

a plurality of data sources, wherein each of the plurality of data sources has a predefined format;

a plurality of agents, wherein each of the plurality of agents is communicatively coupled to one of the plurality of data sources, wherein each of the plurality of agents is configured according to the predefined format of the corresponding data source, and wherein each of the plurality of agents is operative to gather data from the corresponding data source and translate the data into a first modified format thereby creating modified data; and

a reaper communicatively coupled to each of the plurality of agents and configured to retrieve the modified data from each of the plurality of agents.

36. The system of claim 35, further comprising a dashboard controller communicatively coupled to the reaper and configured to display the modified data in a format defined by a configuration parameter.

37. A method for monitoring a business metric in a networked environment, comprising:

coupling to a data source having a known format, wherein the data source includes data that represents the business metric;

configuring an agent according to the data source format;

gathering the data from the data source via the agent;

translating the data into a first modified format;

storing the modified data in a data repository;

comparing the modified data with an alert parameter range;

displaying the modified data in a format defined by a second configuration parameter;

determining whether the modified data falls within the alert parameter range; and

producing an alert if the modified data falls within the alert parameter range.

38. The method of claim 37, wherein the data source is a third party application accessible through a URL address.

39. The method of claim 37, further comprising interfacing with a display device, wherein displaying the modified data in a format defined by a second configuration parameter is implemented on the display device.

40. The method of claim 39, wherein the display device is a cell phone.

41. The method of claim 39, wherein the display device is a pager.

42. The method of claim 39, wherein the display device is a personal computer monitor.

43. A computer-readable medium having computer-executable instructions for performing a method of:

coupling to a data source having a known format, wherein the data source includes data that represents the business metric;

configuring an agent according to the data source format;

gathering the data from the data source via the agent;

translating the data into a first modified format;

storing the modified data in a data repository;

comparing the modified data with an alert parameter range;

displaying the modified data in a format defined by a second configuration parameter;

determining whether the modified data falls within the alert parameter range; and

producing an alert if the modified data falls within the alert parameter range.

44. A method for monitoring a business metric in a networked environment, comprising:

means for coupling to a data source having a known format, wherein the data source includes data that represents the business metric;

means for configuring an agent according to the data source format;

means for gathering the data from the data source via the agent;

means for translating the data into a first modified format;

means for storing the modified data in a data repository;

means for comparing the modified data with an alert parameter range;

means for displaying the modified data in a format defined by a second configuration parameter;

means for determining whether the modified data falls within the alert parameter range; and

means for producing an alert if the modified data falls within the alert parameter range.

45. A system for monitoring a business metric in a networked environment, comprising:

a processor;

a data storage device; and

an instruction set residing on the data storage device, wherein the instruction set is configured to perform a method, the method comprising

coupling to a data source having a known format, wherein the data source includes data that represents the business metric;

configuring an agent according to the data source format;

gathering the data from the data source via the agent;

translating the data into a first modified format;

storing the modified data in a data repository;

comparing the modified data with an alert parameter range;

displaying the modified data in a format defined by a second configuration parameter;

determining whether the modified data falls within the alert parameter range; and

producing an alert if the modified data falls within the alert parameter range.

46. A system for monitoring business performance indicators in a networked environment, comprising:

a data source having a predefined format;

a collector communicatively coupled to the data source, wherein the collector is configured according to the data source format and wherein the collector is operative to gather data from the data source and translate the data into a first modified format thereby creating modified data;

a controller communicatively coupled to the collector and configured to retrieve the modified data from the collector;

a storage device communicatively coupled to the controller and configured to store the modified data;

an alert detector communicatively coupled to the storage device and configured to compare the modified data with a first configuration parameter; and

a display interface communicatively coupled to the storage device and configured to display the modified data in a visual dashboard format defined by a second configuration parameter.

47. A system for monitoring business performance indicators in a networked environment, comprising:

a collector adapted to communicatively coupled to a data source having a predetermined format, wherein the collector is configured according to the data source format and wherein the collector is operative to gather data from the data source and translate the data into a first modified format thereby creating modified data;

a data manager communicatively coupled to the collector and configured to manage the input and output of the modified data between the collector and a

data storage device, wherein the data manager is adapted to communicatively couple with an alert device; and

a display interface communicatively coupled to the data manager and configured to display the modified data in a format defined by a second configuration parameter.

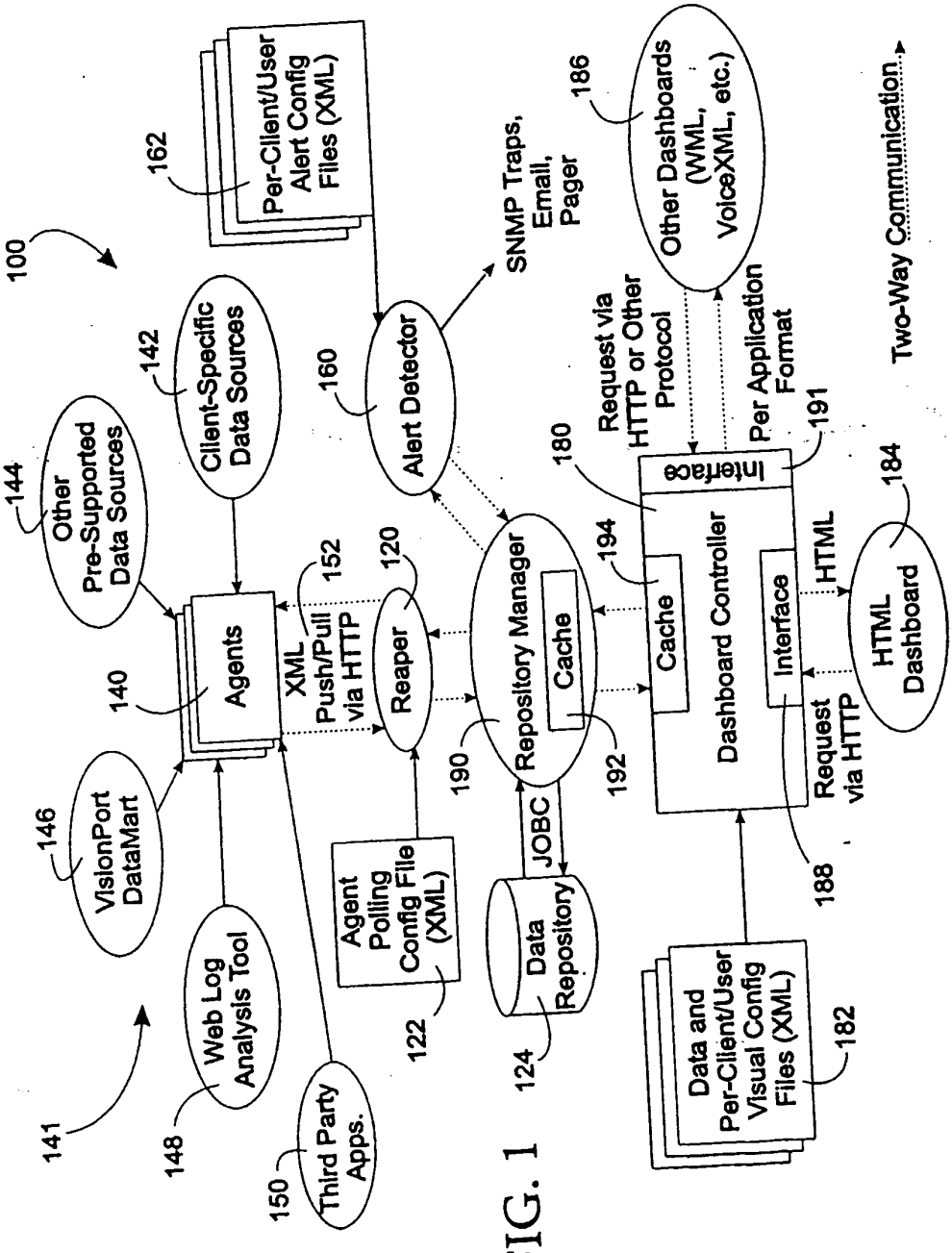

100
162
Per-Client/User Alert Config Files (XML)
142
Client-Specific Data Sources
144
Other Pre-Supported Data Sources
140
Agents
146
VisionPort DataMart
141
148
Web Log Analysis Tool
150
Third Party Apps.
152
XML Push/Pull via HTTP
120
Reaper
122
Agent Polling Config File (XML)
160
Alert Detector
SNMP Traps, Email, Pager
190
Repository Manager
Cache
JOBC
124
Data Repository
192
180
194
Cache
Dashboard Controller
Interface
191
Interface
Request via HTTP or Other Protocol
Per Application Format
186
Other Dashboards (WML, VoiceXML, etc.)
HTML
184
HTML Dashboard
Two-Way Communication
188
Request via HTTP
182
Data and Per-Client/User Visual Config Files (XML)

FIG. 1

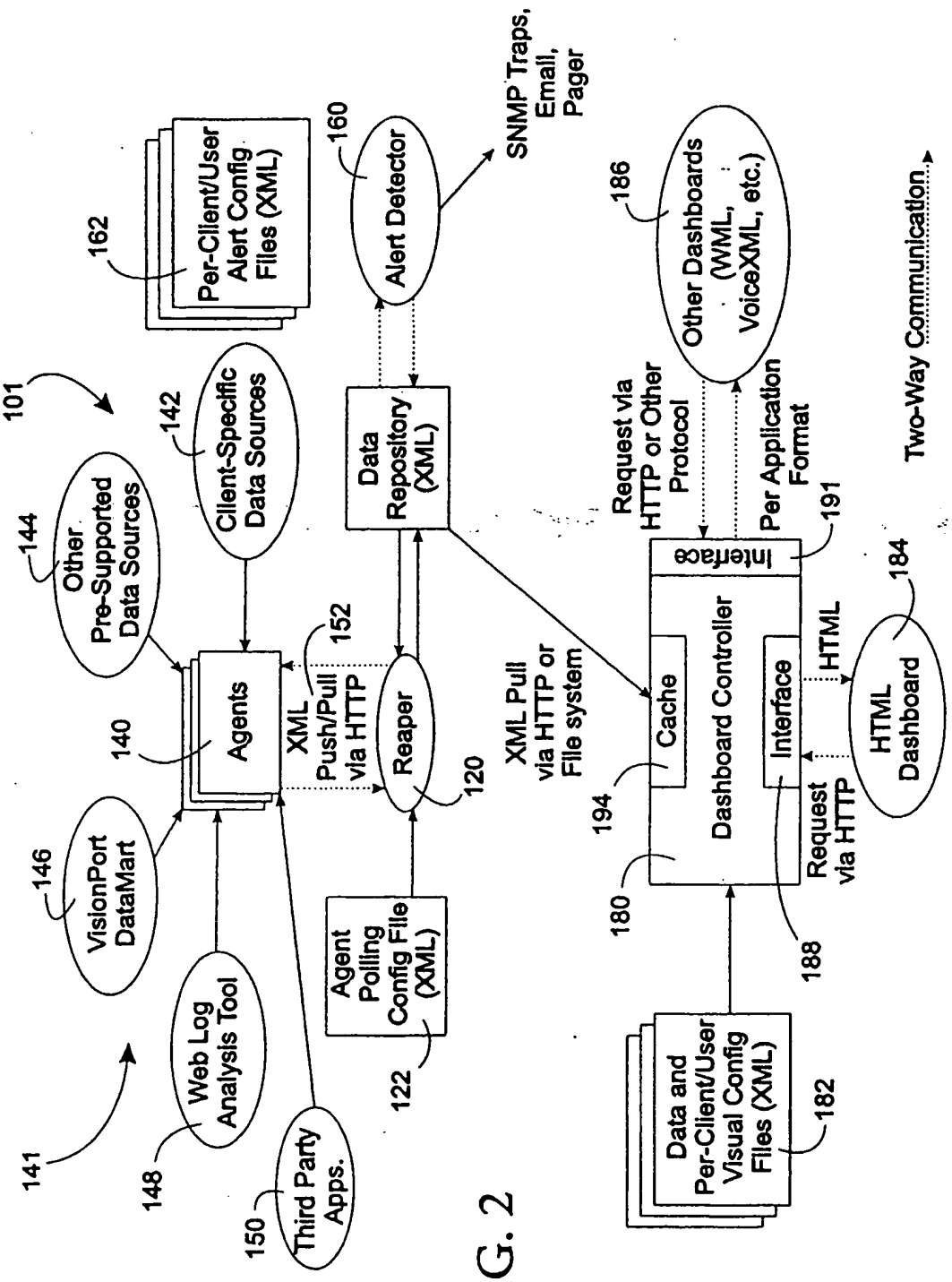

101
141
140
Agents
142
Client-Specific Data Sources
144
Other Pre-Supported Data Sources
146
VisionPort DataMart
148
Web Log Analysis Tool
150
Third Party Apps.
152
XML Push/Pull via HTTP
120
Reaper
122
Agent Polling Config File (XML)
160
Alert Detector
162
Per-Client/User Alert Config Files (XML)
SNMP Traps, Email, Pager
Data Repository (XML)
XML Pull via HTTP or File system
180
Dashboard Controller
194
Cache
191
Interface
Request via HTTP or Other Protocol
Per Application Format
186
Other Dashboards (WML, VoiceXML, etc.)
HTML
184
HTML Dashboard
Request via HTTP
188
182
Data and Per-Client/User Visual Config Files (XML)
Two-Way Communication

FIG. 2

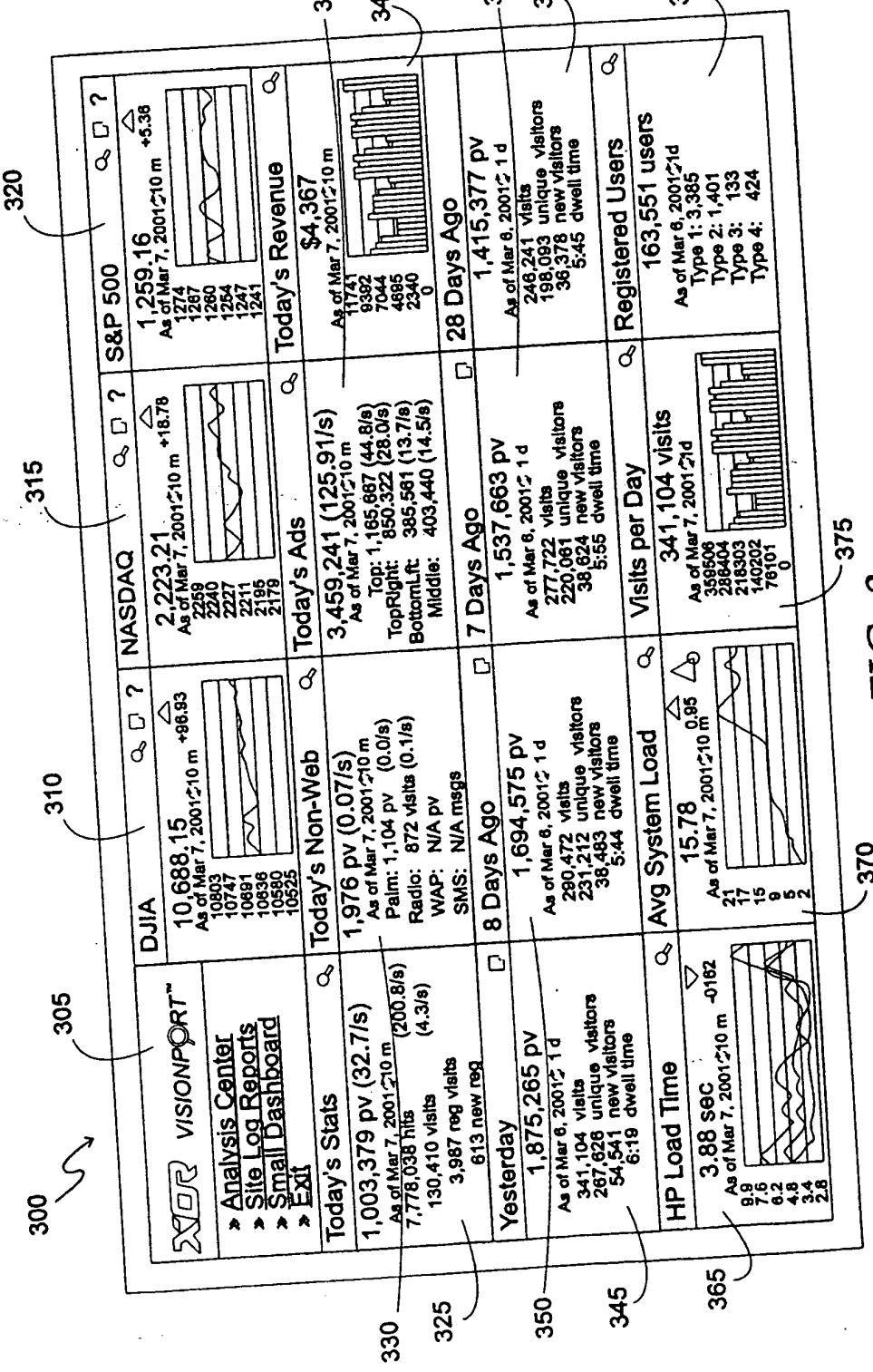

300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
XOR VISIONPORT™
» Analysis Center
» Site Log Reports
» Small Dashboard
» Exit
DJIA
10,688.15
As of Mar 7, 2001 10 m
+96.93
10803
10747
10691
10636
10580
10525
NASDAQ
2,223.21
As of Mar 7, 2001 10 m
+18.78
2259
2240
2227
2211
2195
2179
S&P 500
1,259.16
As of Mar 7, 2001 10 m
+5.38
1274
1267
1260
1254
1247
1241
Today's Stats
1,003,379 pv (32.7/s)
As of Mar 7, 2001 10 m
7,778,038 hits (200.8/s)
130,410 visits (4.3/s)
3,987 reg visits
613 new reg
Today's Non-Web
1,976 pv (0.07/s)
As of Mar 7, 2001 10 m
Palm: 1,104 pv (0.0/s)
Radio: 872 visits (0.1/s)
WAP: N/A pv
SMS: N/A msgs
Today's Ads
3,459,241 (125.91/s)
As of Mar 7, 2001 10 m
Top: 1,165,667 (44.8/s)
TopRight: 850,322 (28.0/s)
BottomLft: 385,581 (13.7/s)
Middle: 403,440 (14.5/s)
Today's Revenue
$4,367
As of Mar 7, 2001 10 m
11741
9392
7044
4695
2340
0
Yesterday
1,875,265 pv
As of Mar 6, 2001 1 d
341,104 visits
267,626 unique visitors
54,541 new visitors
6:19 dwell time
8 Days Ago
1,694,575 pv
As of Mar 6, 2001 1 d
290,472 visits
231,212 unique visitors
38,483 new visitors
5:44 dwell time
7 Days Ago
1,537,663 pv
As of Mar 6, 2001 1 d
277,722 visits
220,061 unique visitors
38,624 new visitors
5:55 dwell time
28 Days Ago
1,415,377 pv
As of Mar 6, 2001 1 d
246,241 visits
198,093 unique visitors
36,378 new visitors
5:45 dwell time
HP Load Time
3.88 sec
As of Mar 7, 2001 10 m
-0162
9.9
7.6
6.2
4.8
3.4
2.8
Avg System Load
15.78
As of Mar 7, 2001 10 m
0.95
21
17
15
9
5
2
Visits per Day
341,104 visits
As of Mar 7, 2001 1d
359508
288404
218303
140202
78101
0
Registered Users
163,551 users
As of Mar 6, 2001 1d
Type 1: 3,385
Type 2: 1,401
Type 3: 133
Type 4: 424

FIG. 3

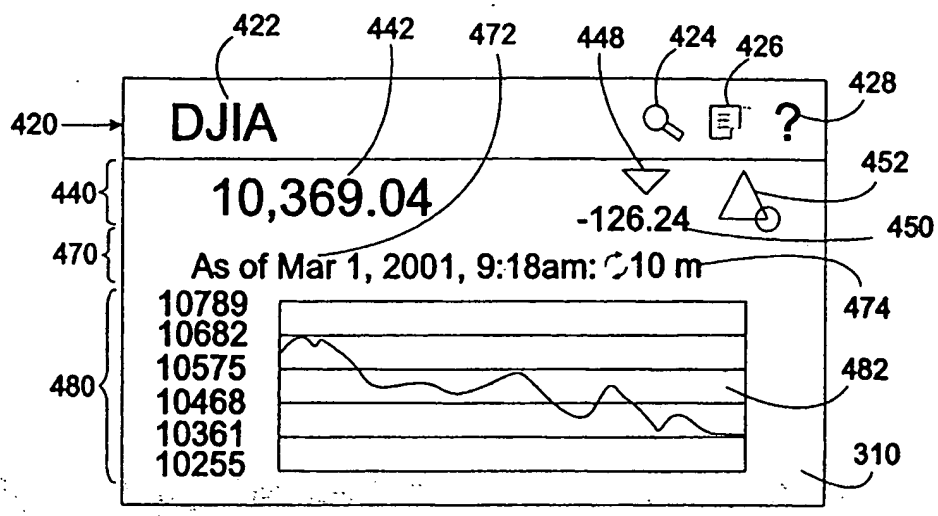

422
442
472
448
424
426
428
420
DJIA
?
452
440
10,369.04
-126.24
450
470
As of Mar 1, 2001, 9:18am: 10 m
474
10789
10682
10575
10468
10361
10255
480
482
310

FIG. 5

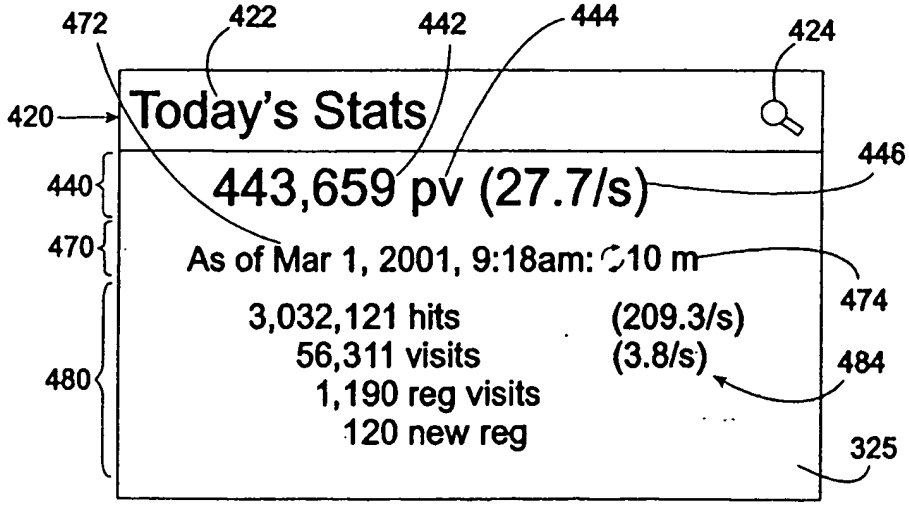

472
422
442
444
424
420
Today's Stats
446
440
443,659 pv (27.7/s)
470
As of Mar 1, 2001, 9:18am: 10 m
474
3,032,121 hits
(209.3/s)
56,311 visits
(3.8/s)
484
480
1,190 reg visits
120 new reg
325

FIG. 6

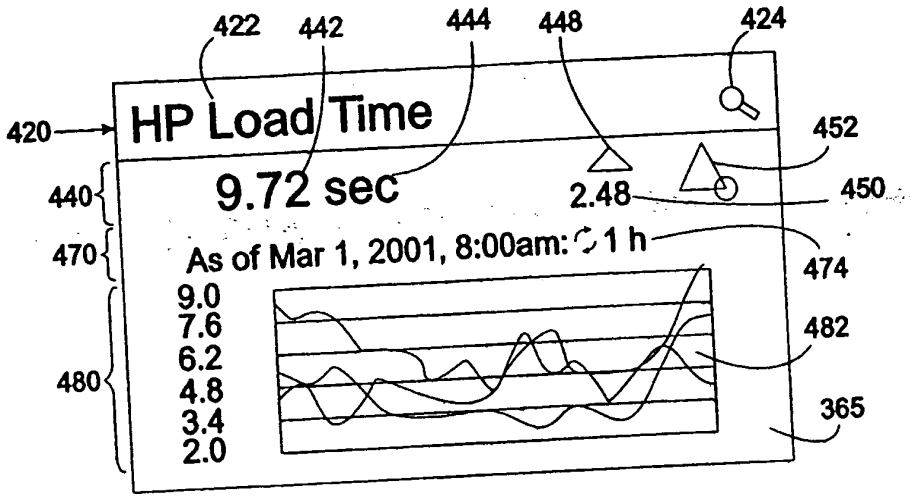

422
442
444
448
424
420
HP Load Time
452
440
9.72 sec
2.48
450
470
As of Mar 1, 2001, 8:00am:
1 h
474
9.0
7.6
6.2
4.8
3.4
2.0
480
482
365

FIG. 7

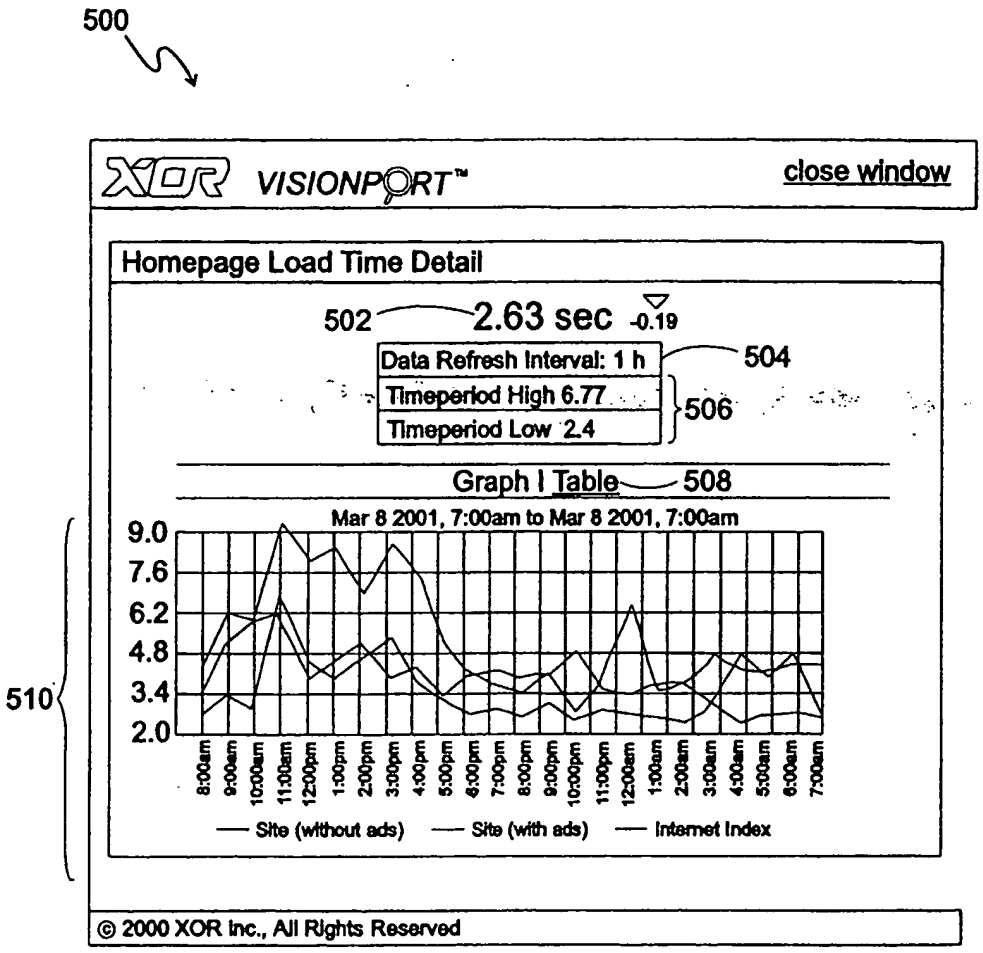
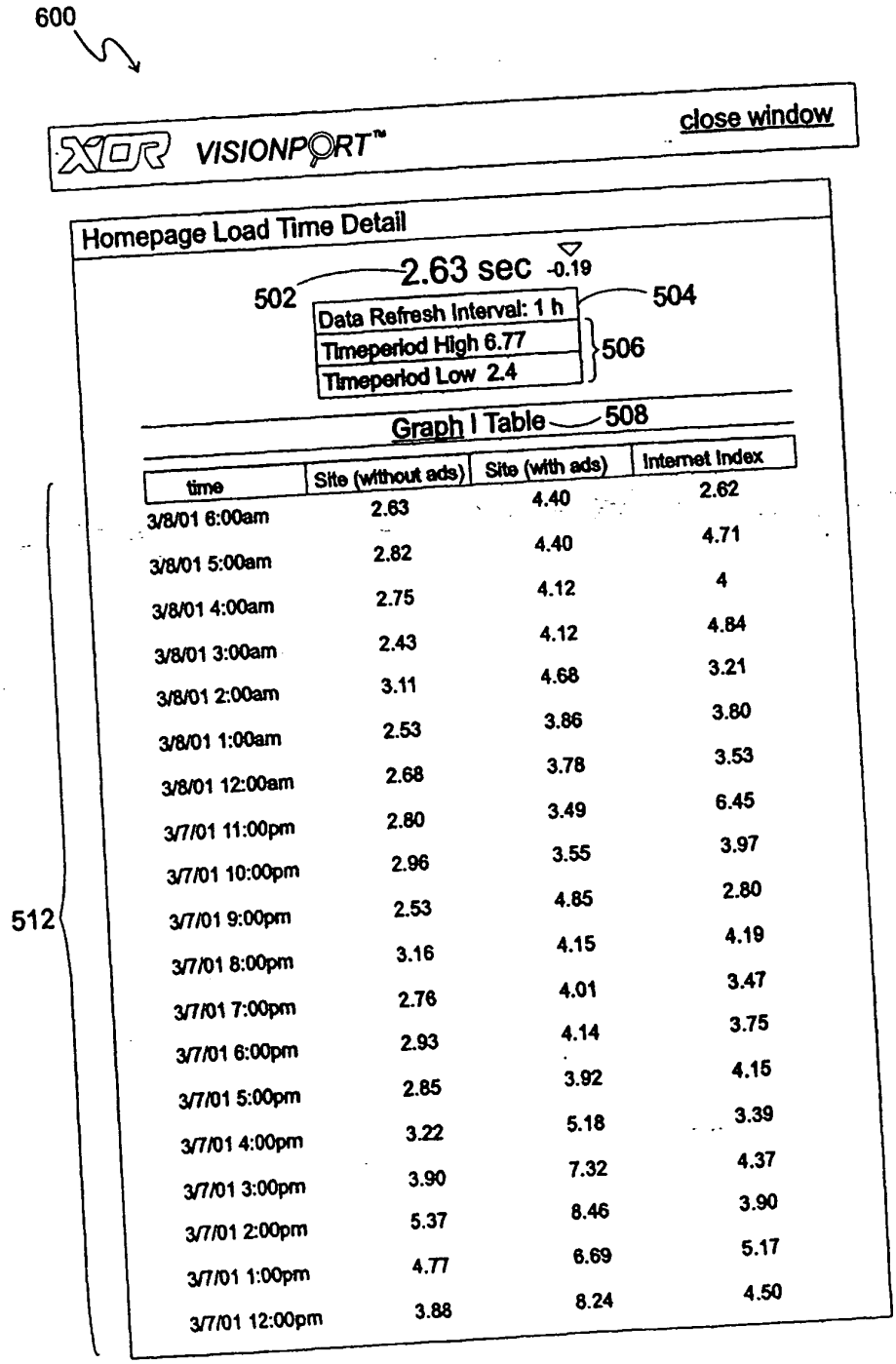

500
XOR VISIONPORT™
close window
Homepage Load Time Detail
502
2.63 sec
-0.19
Data Refresh Interval: 1 h
504
Timeperiod High 6.77
Timeperiod Low 2.4
506
Graph I Table
508
Mar 8 2001, 7:00am to Mar 8 2001, 7:00am
9.0
7.6
6.2
4.8
3.4
2.0
510
8:00am
9:00am
10:00am
11:00am
12:00pm
1:00pm
2:00pm
3:00pm
4:00pm
5:00pm
6:00pm
7:00pm
8:00pm
9:00pm
10:00pm
11:00pm
12:00am
1:00am
2:00am
3:00am
4:00am
5:00am
6:00am
7:00am
Site (without ads)
Site (with ads)
Internet Index
© 2000 XOR Inc., All Rights Reserved

FIG. 8



time	Site (without ads)	Site (with ads)	Internet Index
3/8/01 6:00am	2.63	4.40	2.62
3/8/01 5:00am	2.82	4.40	4.71
3/8/01 4:00am	2.75	4.12	4
3/8/01 3:00am	2.43	4.12	4.84
3/8/01 2:00am	3.11	4.68	3.21
3/8/01 1:00am	2.53	3.86	3.80
3/8/01 12:00am	2.68	3.78	3.53
3/7/01 11:00pm	2.80	3.49	6.45
3/7/01 10:00pm	2.96	3.55	3.97
3/7/01 9:00pm	2.53	4.85	2.80
3/7/01 8:00pm	3.16	4.15	4.19
3/7/01 7:00pm	2.76	4.01	3.47
3/7/01 6:00pm	2.93	4.14	3.75
3/7/01 5:00pm	2.85	3.92	4.15
3/7/01 4:00pm	3.22	5.18	3.39
3/7/01 3:00pm	3.90	7.32	4.37
3/7/01 2:00pm	5.37	8.46	3.90
3/7/01 1:00pm	4.77	6.69	5.17
3/7/01 12:00pm	3.88	8.24	4.50

FIG. 9

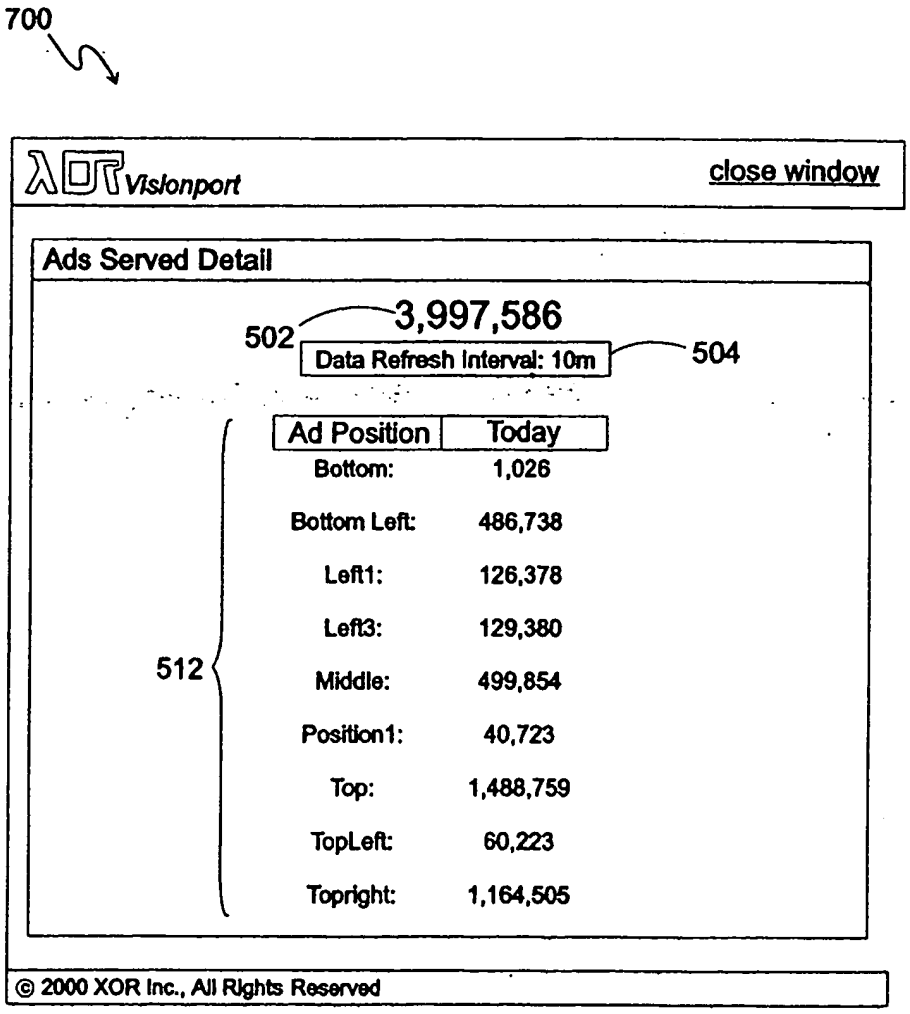

700
XOR Visionport
close window
Ads Served Detail
502
3,997,586
Data Refresh Interval: 10m
504
512
Ad Position
Today
Bottom: 1,026
Bottom Left: 486,738
Left1: 126,378
Left3: 129,380
Middle: 499,854
Position1: 40,723
Top: 1,488,759
TopLeft: 60,223
Topright: 1,164,505
© 2000 XOR Inc., All Rights Reserved

FIG. 10

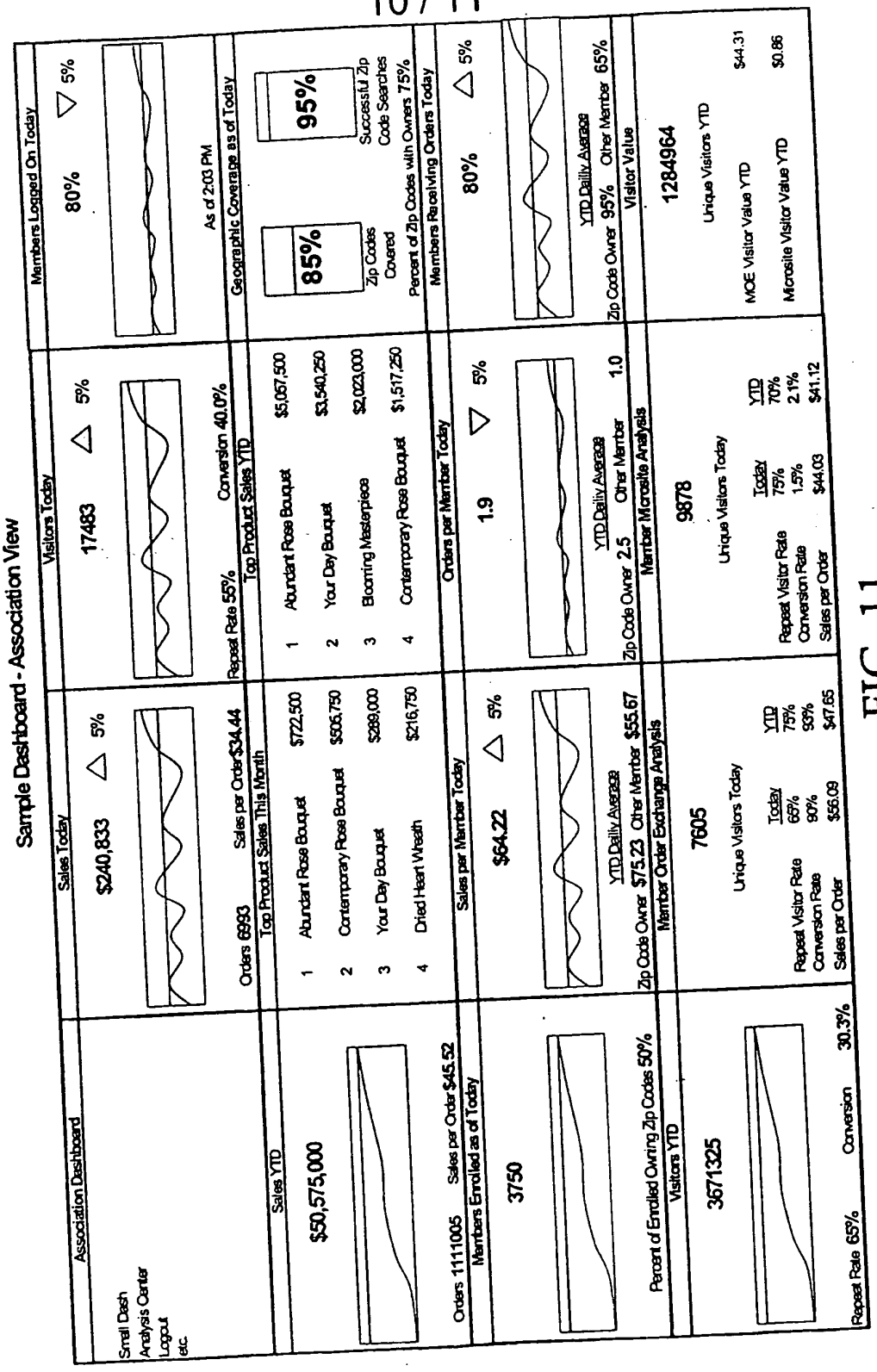

Sample Dashboard - Association View
Association Dashboard
Small Dash
Analysis Center
Logout
etc.
Sales Today
$240,833
5%
Orders 6993
Sales per Order $34.44
Visitors Today
17483
5%
Repeat Rate 55%
Conversion 40.0%
Members Logged On Today
80%
5%
As of 2:03 PM
Geographic Coverage as of Today
85%
Zip Codes Covered
95%
Successful Zip Code Searches
Percent of Zip Codes with Owners 75%
Top Product Sales This Month
1 Abundant Rose Bouquet $722,500
2 Contemporary Rose Bouquet $506,750
3 Your Day Bouquet $289,000
4 Dried Heart Wreath $216,750
Top Product Sales YTD
1 Abundant Rose Bouquet $5,067,500
2 Your Day Bouquet $3,540,250
3 Blooming Masterpiece $2,023,000
4 Contemporary Rose Bouquet $1,517,250
Members Receiving Orders Today
80%
5%
Sales YTD
$50,575,000
Orders 1111005
Sales per Order $45.52
Sales per Member Today
$64.22
5%
YTD Daily Average
Zip Code Owner $75.23 Other Member $55.67
Orders per Member Today
1.9
5%
YTD Daily Average
Zip Code Owner 2.5 Other Member 1.0
YTD Daily Average
Zip Code Owner 95% Other Member 65%
Visitor Value
Members Enrolled as of Today
3750
Percent of Enrolled Owning Zip Codes 50%
Member Order Exchange Analysis
Member Microsite Analysis
Visitors YTD
3671325
Repeat Rate 65%
Conversion 30.3%
7605
Unique Visitors Today
Today YTD
Repeat Visitor Rate 69% 75%
Conversion Rate 90% 93%
Sales per Order $56.09 $47.65
9878
Unique Visitors Today
Today YTD
Repeat Visitor Rate 75% 70%
Conversion Rate 1.5% 2.1%
Sales per Order $44.03 $41.12
1284964
Unique Visitors YTD
MOE Visitor Value YTD $44.31
Microsite Visitor Value YTD $0.86

FIG. 11

Sample Dashboard - Franchise View

Franchise Dashboard	Sales Today	Orders Today	Visitors Today
Small Dash Analysis Center Logout etc.	$212.45 △ 5%	5 △ 5%	155 ▽ 5%
	Sales per Order $45.70	Conversion Rate 3.0%	Repeat Rate 55%

Sales YTD	Backlog Sales as of Today	Top Product Sales This Month			Top Product Sales YTD		
$44,614.50	$6,692.18	1	Abundant Rose Bouquet	$637	1	Abundant Rose Bouquet	$4,461
	0 to 15 Days $3,346.09	2	Contemporary Rose Bouquet	$446	2	Your Day Bouquet	$3,123
	15 to 30 Days $2,007.65	3	Your Day Bouquet	$255	3	Blooming Masterpiece	$1,785
Orders 1109 Sales per Order $40.23	Over 30 Days $1,338.44	4	Dried Heart Wreath	$191	4	Contemporary Rose Bouquet	$1,338

Microsite Sales YTD	Microsite Orders YTD	Microsite Visitor Value	Microsite Visitor Behavior		
$11,153.63	311	10351		Today	YTD
		Unique Visitors YTD	Session length	0:13:45	0:14:15
		Value per unique visitor YTD $1.08	Pageviews	12	13
Sales per Order $35.92	Repeat Rate 65%	Marketing costs per unique visitor YTD $0.97	Cart abandonment rate	10%	12%

Member Order Exchange Sales YTD	Member Order Exchange Orders YTD	Top Referring Zip Codes YTD			Top Referring Florists YTD		
$33,460.88	798	1	48858	$8,365.22	1	Norm's Flower Petal	$4,182.61
		2	60618	$3,346.09	2	Roscoe Stems	$1,673.04
		3	55590	$2,342.26	3	Flowers By Bob	$1,171.13
Percent Referred to Me 55%	Percent Referred to Me 52%	4	34112	$1,673.04	4	The New Leaf	$836.52

FIG. 12

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/18069

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :H04Q 7/20; G06F 17/60
US CL : 705/35; 705/36

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 705/35; 705/36

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EAST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6,240,295 B1 (KENNEDY, III et. al.) 29 MAY 2001, Abstract; Figs. 1-11; entire document.	1-47
Y	US 6,119,103 A (BASCH et al.) 12 SEPTEMBER 2000, Abstract; Figs 1-9; entire document.	1-47

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
17 SEPTEMBER 2002

Date of mailing of the international search report
29 OCT 2002

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703) 305-3230

Authorized officer
GEOFFREY AKERS

Telephone No. (703) -308-5844

Form PCT/ISA/210 (second sheet) (July 1998)*